AD-A061 729 ACADEMIC COMPUTER CENTER UTRECHT (NETHERLANDS) F/G 12/1
FAST ITERATIVE METHODS FOR LARGE LINEAR SYSTEMS.(U)
AUG 78 H A VORST DA-ERO-75-G-0084

UNCLASSIFIED NL

1 OF 2
AD
A061729

FAST ITERATIVE METHODS FOR LARGE LINEAR SYSTEMS

Final Technical Report

by

H.A. van der Vorst

August 1978

European Research Office
United States Army
London England

GRANT NUMBER DA-ERO-75-G-084

Academic Computer Center, Utrecht.

AD

FAST ITERATIVE METHODS FOR LARGE LINEAR SYSTEMS

Final Technical Report

by

H.A. van der Vorst

August 1978

European Research Office
United States Army
London England

GRANT NUMBER DA-ERO-75-G-084

Academic Computer Center, Utrecht.

Approved for Public Release; distribution unlimited.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)

(1) East Iterative Methods for Large Linear Systems,

5. TYPE OF REPORT & PERIOD COVERED

(9) Final Technical Report,

6. ~~PERFORMING ORG. REPORT NUMBER~~

7. AUTHOR(s)

H.A. van der Vorst (10) Henk A. van der Vorst

8. CONTRACT OR GRANT NUMBER(s)

(15) DA-ERO-75-G-0084

9. PERFORMING ORGANIZATION NAME AND ADDRESS

Academic Computer Center, Utrecht, The Netherlands 392 112

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

6.11.02A (16) 1T161102B18C
(17) 00 538

11. CONTROLLING OFFICE NAME AND ADDRESS

US ARMY R&S GP (EUROPE)
BOX 65, New York NY 09510

12. ~~REPORT DATE~~

(11) Aug 78

13. NUMBER OF PAGES

101

14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)

AS ABOVE

15. SECURITY CLASS. (of this report)

UNCLASSIFIED

15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION ~~STATEMENT (of this Report)~~

(12) 115p.

~~Approved for Public~~ Release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Fast Iterative Methods, Sparse Matrices, Partial Differential Equations.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

In reference 1 of the report, special splittings of the matrix of a sparse linear system were proposed. These splittings can be combined with the conjugate gradients method, which results in very efficient iterative solution methods when the matrix is a symmetric M-matrix. The research reported in this Final Technical Report has been focused on three major subjects: → next page

SEE BACK

392 112

(a) a systematic overview of possible splittings for several types of matrices and eigenvalue information.

(b) applications to nonsymmetric linear systems. and

(c) solution of generalized eigenvalue problems for sparse linear systems.

The research on (a) has resulted in more insight on how to choose the appropriate preconditioning for matrices of a special structure. Moreover, splittings have been proposed for other problems, e.g. 3D-problems and periodic boundary condition problems.

The research on (b) has not yet led to satisfactory iterative solution methods, though for special problems an always converging method has been developed.

The eigenvalue problems mentioned under (c) have been solved satisfactorily with Lanczos-type algorithms. These problems were met in the investigation of convergence properties of the methods mentioned under (a) and (b).

Summary. In ref. 1 special splittings of the matrix of a sparse linear system were proposed. These splittings can be combined with the conjugate gradients method, which results in very efficient iterative solution methods when the matrix is a symmetric M-matrix [1].

The research reported in this Final Technical Report has been focused on three major subjects:

(a) a systematic overview of possible splittings for several types of matrices and eigenvalue information

(b) applications to nonsymmetric linear systems

(c) solution of generalized eigenvalue problems for sparse linear systems.

The research on (a) has resulted in more insight on how to choose the appropriate preconditioning for matrices of a special structure. Moreover, splittings have been proposed for other problems, e.g. 3D-problems and periodic boundary condition problems.

The research on (b) has not yet led to satisfactory iterative solution methods, though for special problems an always converging method has been developed.

The eigenvalue problems mentioned under (c) have been solved satisfactorily with Lanczos-type algorithms. These problems were met in the investigation of convergence properties of the methods mentioned under (a) and (b).

[1] A matrix $A=(a_{ij})$ is an M-matrix if $a_{ij}\le 0$ for $i\neq j$, A is nonsingular and $A^{-1}\ge 0$.

Table of Contents

I. Splitting techniques for several types of symmetric matrices

In ref. 1 the idea of incomplete decompositions LU for arbitrary M-matrices A was introduced. For symmetric matrices A those decompositions were used as preconditionings for the conjugate gradient algorithm to solve the linear system Ax=b. In the examples given in ref. 1 matrices were considered arising from 5-point discretisation of a self-adjoint elliptic partial differential equation on a rectangular region.

During the period of the Grant, attention has been given to a more systematic investigation of possible preconditionings. From the information that came out this investigation also precoditionings could be proposed for other types of matrices. We mention here problems that come from p.d.e.'s with periodic boundary conditions, 3D-problems and problems where the resulting matrix has an arbitrary non-zero structure. Also research has been done for problems where the matrix is not an M-matrix.

This research has been reported briefly in ref. 2 and extensively in Appendix A. As a conclusion we mention that, with respect to either economy of space or the amounts of computational work, optimal choices for a preconditioning can be made for problems that come from discretisations of elliptic self-adjoint p.d.e.'s.

The eigenvalue information, reported in Appendix A, demonstrates the effects of constructing a more or less incomplete decomposition for use as a preconditioning.

II. Iterative methods for nonsymmetric systems

The algorithms mentioned in section I work all fine for symmetric positive definite linear systems but they fail to work for nonsymmetric systems since they are based on the conjugate gradients method. Variants of the conjugate gradients algorithm, that converge also for certain classes of nonsymmetric systems, are proposed by Concus and Golub [3], Widlund [4] and others. They share the disadvantage that a symmetric system has to be solved in each iteration step. Thus in general these methods can not compete with the original cg-method for symmetric matrices.

It has been investigated during the Grant-period whether incomplete LU-decompositions could be used to speed up the convergence of some well-chosen method. Two different ways have been followed, each of which seems to have promises but research is only in its very first stage.

The first way was to base iterative methods on the following splitting of a given matrix A:

$$\begin{aligned} A &= \alpha(A + A^T) + ((1 - \alpha)A - \alpha A^T) \\ &= \alpha M + N \end{aligned}$$

instead of $A=\frac{1}{2}(A+A^T)+\frac{1}{2}(A-A^T)$ which is more usual.

In Appendix B formulas are given that yield a converging method if $A+A^T$ is symmetric and positive definite. It is shown in Appendix B that the replacement of M by its incomplete LL^T-decomposition in general results in faster convergence.

The other way was to follow ideas expressed by Manteuffel [5] based on Chebychev acceleration of iterative methods for non-linear systems. It was recognized that the use of incomplete LU-decompositions of A could be used as a preconditioning and they gave more efficient solution methods than for instance the use of a Fast-Poisson-Solver as a preconditioning.

III. Generalized Eigenvalue Problems

In order to study the convergence properties of the iterative methods mentioned in I and II it was necessary to inspect the eigenvalue distribution of the respective iteration matrices.

The Lanczos-algorithm as proposed by Paige [6] has been chosen to compute a large number of eigenvalues. In ref. 7 the numerical experiments are described for calculating eigenvalues of BA where B and A are both symmetric and positive definite.

Since also eigenvalues were required of matrices BC, where B is symmetric positive definite and $C=-C^T$ (antisymmetric), a variant of the Lanczos-algorithm is proposed in Appendix C.

The often very tedious examination of the numerical results, as usually required when using Lanczos-methods, has been largely overcome by an automatic selection procedure. This research is also described in Appendix C.

Literature Cited

1. Meijerink, J.A. and van der Vorst, H.A., An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix,
   Math. of Comp., Vol. 31 (1977), pp. 148-162.

2. van der Vorst, H.A., Approximate decompositions of sparse matrices, Proceedings of the 1977 Army Numerical Analysis and Computers Conference, ARO Report 77-3 (1977).

3. Concus, P. and Golub, G.H., A generalized Conjugate Gradient Method for Nonsymmetric Systems of Linear Equations,
   Proc. 2nd Int. Symp. on Computing Methods in Applied Sciences and Engineering, IRIA, Paris (1975).

4. Widlund, O., A Lanczos Method for a Class of Non-Hermitian Systems of Linear Equations,
   Courant Institute, New York University (1976).

5. Manteuffel, T.A., Adaptive Procedure for Estimating Parameters for the Nonsymmetric Tchebychev Iteration,
   Report SAND 77-8239, Sandia Laboratories, Livermore (1977).

6. Paige, C.C., Computational Variants of the Lanczos Method for the Eigenproblem,
   J. Inst. Math. Applics. 10 (1972), pp. 373-381.

7. van Kats, J.M. and van der Vorst, H.A., Numerical results of the Paige-style Lanczos method for the computation of extreme eigenvalues of large sparse matrices,
   TR-3, Academic Computer Center Utrecht (1977).

APPENDIX A

Guide lines for the usage of incomplete decompositions in solving sets of linear equations as occur in practical problems.

by

J.A. Meijerink [+] and H.A. van der Vorst.

[+] Koninklijke Shell Exploratie en Produktie Laboratorium
Rijswijk, The Netherlands.

CONTENTS

The two test problems were:

SUMMARY

This report presents incomplete decompositions for various types of matrices as occur in the implicit discretisation of practical problems. A review is given of methods for the usual five-point discretisation of a self-adjoint elliptic second-order partial differential equation in two dimensions on a square. The matrices which occur in this type of problem are symmetric M-matrices of very regular structure. The convergence behaviour of the different decompositions for this case is demonstrated by numerical experiments. The report also gives decompositions for the following type of matrices:

(i) Symmetric M-matrices of a different structure
(ii) Symmetric positive definite matrices
(iii) Non-symmetric positive definite matrices

KEYWORDS

Matrix algebra, numerical analysis, partial differential equation.

GUIDE LINES FOR THE USAGE OF INCOMPLETE DECOMPOSITIONS IN SOLVING SETS OF LINEAR EQUATIONS AS OCCUR IN PRACTICAL PROBLEMS

1. INTRODUCTION

In Ref. 1 the idea of incomplete decomposition LU for arbitrary M-matrices A was introduced. For symmetric matrices A those decompositions were used as preconditionings for the conjugate gradient algorithm to solve the linear equation Ax=b. In the examples given in Ref. 1 matrices were considered arising from 5-point discretisation of a self-adjoint elliptic partial differential equation on a rectangular region. Only two different incomplete decompositions were demonstrated.

In this report we present a more systematic review of the possible incomplete decompositions for that problem (section 3.1). In addition, we shall discuss incomplete decompositions for other types of matrices, e.g. M-matrices arising from problems with periodic boundary conditions (section 3.2) and M-matrices with a more arbitrary structure (sections 3.4 and 3.5). For this purpose we need an extension of the definition of an incomplete decomposition given in the proof of theorem 2.3 in Ref. 1. In the process defined there some off-diagonal elements were omitted after each elimination step. If, instead of omitting some off-diagonal elements, we replace them by negative elements which are smaller in absolute value, or if diagonal elements are replaced by larger ones, the construction process does not break down and the resulting incomplete decomposition 'LU' defines a regular splitting of A. This new process describes the extended concept of incomplete decompositions. A specific example of this process is the following: In the k^{th} step of the Gaussian elimination process elements are eliminated with the k^{th} row. This may cause three effects:

i) zero off-diagonal elements turn to negative non-zero values

ii) non-zero off-diagonal elements become smaller (although larger in absolute value)

iii)diagonal elements become smaller (but remain positive).

Thus omitting to carry out the elimination corrections for some of the elements of the matrix results in an incomplete decomposition. Examples of this type of decomposition are given in sections 2.1.2, 2.1.3, 2.4 and 2.5. Other approximate factorisations are discussed by Stone[2], Dupont et all[3] and

Gustaffson[4]. They all introduce one or more parameters into the decomposition process to accelerate convergence. The property of regular splitting is lost in most of their examples.

Kershaw[5] and Manteuffel[6] provide extensions for positive definite matrices. We shall describe these briefly and present another one in section 3. Incomplete decompositions for p.d.e.'s in three-dimensions are treated in section 2.3. In section 4 algorithms for non-symmetric matrices are described. In section 5 convergence results as well as 'eigenvalue' information on the decompositions described in section 2.1 are given for some specific examples.

2. INCOMPLETE DECOMPOSITIONS FOR SYMMETRIC M-MATRICES

2.1 Five-point discretisation of elliptic partial differential equations in two-dimensions

The linear equations in this section arise from five-point discrete approximation to the second-order self adjoint elliptic partial differential equation:

$$-\frac{\partial}{\partial x} A(x,y) \frac{\partial}{\partial x} u(x,y) - \frac{\partial}{\partial y} B(x,y) \frac{\partial}{\partial y} u(x,y) + C(x,y)\, u(x,y) = D(x,y) \tag{2.1.1}$$

with $A(x,y)$, $B(x,y)>0$, $C(x,y)\geq 0$ and $(x,y) \in R$, where R is a rectangular region. Along the boundary δR of R the boundary condition

$$\alpha(x,y)\, u(x,y) + \beta(x,y) \frac{\partial}{\partial n} u(x,y) = \gamma(x,y)$$

holds, with $\alpha(x,y)$, $\beta(x,y)\geq 0$ and $\alpha(x,y) + \beta(x,y)>0$ and where $\frac{\partial}{\partial n}$ is the outward derivative perpendicular to δR. The structure of the resulting symmetric M-matrix A of order N is shown in Fig. 1. The elements of the diagonal of A are denoted by a_i, those of the first upper diagonal by b_i and those of the m-th upper diagonal by c_i, where i is the index of the row of A in which the respective elements occur, and m is the half bandwidth of the matrix. For the derivation of such linear systems see Ref. 7.

2.1.1 Diagonal scaling

The simplest allowed choice for K is the diagonal of A. The resulting conjugate gradient method is the same as the c.g. method applied on the matrix scaled by its diagonal. This scaling is in some respect optimal, since it minimises approximately the condition number of $K^{-1}A$ among all diagonal scalings[8].

If the equation is scaled in advance, the number of multiplications is 10 N per iteration. If not, this number will be 11 N.

2.1.2 ICCG (1,1) and SSOR (ω=1)

Here the matrix K is chosen so that its decomposition factor L^T has the same sparsity pattern as the upper triangular part of A. This decomposition has been considered by many authors[1,2,3,4,9].

It is convenient to write this decomposition as $K_{1,1} = L_{1,1} D_{1,1} L^T_{1,1}$, where $L^T_{1,1}$ is an upper triangular matrix and $D_{1,1}$ a diagonal matrix equal to the inverse of the main diagonal of $L^T_{1,1}$. In common with the elements of A, those of $L^T_{1,1}$ are denoted by $\tilde{a}_i$ $\tilde{b}_i$ and $\tilde{c}_1$ and those of $D_{1,1}$ by $\tilde{d}_i$ (see Fig. 2). The following relations are easily verified:

$$\tilde{a}_i = \tilde{d}_i^{-1} = a_i - \tilde{b}_{i-1}^2 \tilde{d}_{i-1} - \tilde{c}_{i-m}^2 \tilde{d}_{i-m}$$

$$\text{for } i=1,2,\ldots,N$$

$$\tilde{b}_i = b_i \text{ and } \tilde{c}_i = c_i$$

In these relations the non-defined elements are zero. Only extra storage for the elements $\tilde{d}_i$ is required. The resulting hybrid conjugate gradient method is called ICCG(1,1). The indices are used to indicate that there is one non-zero diagonal next to the main diagonal and one non-zero diagonal at the outward side of the band. This is ICCG(0) of Ref. 1. The number of multiplications for this method is 16 N multiplications per iteration.

The SSOR(ω=1) decomposition arises if all Gaussian elimination corrections are neglected. Thus SSOR(ω=1) is an example of the extended class of incomplete decompositions mentioned in section 1. The number of multiplications remains the same as for ICCG(1,1), but one array of storage has been saved. For the use of SSOR as a preconditioning technique see [10].

2.1.3 ICCG(1,2)

The matrix $K_{1,1} = L_{1,1} D_{1,1} L_{1,1}$ of the previous section is a matrix equal to A, except for two diagonals adjacent to the outermost two diagonals, as indicated by the dotted lines in Fig. 3. By including non-zero entries on those lines in L and L^T, we expect to improve the approximate decomposition. This approximation will be written as

$$K_{1,2} = L_{1,2} D_{1,2} L_{1,2}^T$$

where $D_{1,2}$ is the diagonal matrix equal to the inverse of the main diagonal of $L_{1,2}^T$. The elements of $L_{1,2}^T$ are denoted by $\tilde{a}_i$, $\tilde{b}_i$, $\tilde{e}_i$ and $\tilde{c}_i$ and those of $D_{1,2}$ by $\tilde{d}_i$ (see Fig. 4). The elements $\tilde{a}_i$, $\tilde{b}_i$, $\tilde{c}_i$, $\tilde{d}_i$ and $\tilde{e}_i$ can be computed recursively from

$$\begin{aligned}
\tilde{a}_i = \tilde{d}_i^{-1} &= a_i - \tilde{b}_{i-1}^2 \tilde{d}_{i-1} - \tilde{e}_{i-m+1}^2 \tilde{d}_{i-m+1} - \tilde{c}_{i-m}^2 \tilde{d}_{i-m} \\
\tilde{b}_i &= b_i - \tilde{c}_{i-m+1} \tilde{d}_{i-m+1} \tilde{e}_{i-m+1} \qquad \text{for } i=1,2,\ldots,N \\
\tilde{e}_i &= - \tilde{c}_{i-1} \tilde{d}_{i-1} \tilde{b}_{i-1} \\
\tilde{c}_i &= c_i
\end{aligned}$$

The non-defined elements are all zero.

Storage is required for three arrays of length N. The number of multiplications necessary for each iteration step of the resulting hybrid conjugate gradient method ICCG(1,2) is equal to 18 N.

In order to save computer storage (one array) the relatively small correction on $\tilde{b}_i$ can be omitted. Thus $\tilde{b}_i = b_i$. This is another example of the extended class of incomplete decompositions and will be denoted by ICCG $(1^*,2)$

2.1.4 ICCG(1,3)

The matrix $K_{1,2}$ is equal to A, except for the two dotted diagonals as indicated in Fig. 5. These non-zero elements can be eliminated by introducing an extra diagonal in L^T (see Fig. 6). The elements on these diagonals will be denoted by $\tilde{f}_i$. This incomplete decomposition will be denoted by

$$K_{1,3} = L_{1,3} D_{1,3} L_{1,3}^T$$

The elements of $D_{1,3}$ and $L_{1,3}^{T}$ can be computed from:

$$\begin{aligned}
\tilde{a}_i &= \tilde{d}_i^{-1} = a_i - \tilde{b}_{i-1}^2 \tilde{d}_{i-1} - \tilde{f}_{i-m+2}^2 \tilde{d}_{i-m+2} - \tilde{e}_{i-m+1} \tilde{d}_{i-m+1} - \tilde{c}_{i-m}^2 \tilde{d}_{i-m} \\
\tilde{b}_i &= b_i - \tilde{c}_{i-m+1} \tilde{d}_{i-m+1} \tilde{e}_{i-m+1} - \tilde{e}_{i-m+2} \tilde{d}_{i-m+2} \tilde{f}_{i-m+2} \\
\tilde{f}_i &= -\tilde{e}_{i-1} \tilde{d}_{i-1} \tilde{b}_{i-1} & i=1,2,\dots,N \\
\tilde{e}_i &= -\tilde{c}_{i-1} \tilde{d}_{i-1} \tilde{b}_{i-1} \\
\tilde{c}_i &= c_i
\end{aligned}$$

The non-defined elements are zero.

The resulting ICCG(1,3) process necessitates about 20 N multiplications per iteration and requires four arrays of length N for the decomposition.

2.1.5 ICCG(2,4)

Unlike $K_{1,1}$ and $K_{1,2}$, $K_{1,3}$ differs from A by four non-zero diagonals (Fig. 7). To eliminate these, two more non-zero diagonals in L^T are necessary. The elements on these diagonals are denoted by $\tilde{h}_i$ and $\tilde{g}_i$ (Fig. 8). This incomplete decomposition is written as

$$K_{2,4} = L_{2,4} D_{2,4} L_{2,4}^{T}$$

and the elements of $L_{2,4}^{T}$ and $D_{2,4}$ follow from

$$\begin{aligned}
\tilde{a}_i &= \tilde{d}_i^{-1} = a_i - \tilde{b}_{i-1}^2 \tilde{d}_{i-1} - \tilde{h}_{i-2}^2 \tilde{d}_{i-2} - \tilde{g}_{i-m+3}^2 \tilde{d}_{i-m+3} - \tilde{f}_{i-m+2}^2 \tilde{d}_{i-m+2} - \tilde{e}_{i-m+1}^2 \tilde{d}_{i-m+1} - \tilde{c}_{i-m}^2 \tilde{d}_{i-m} \\
\tilde{b}_i &= b_i - \tilde{h}_{i-1} \tilde{d}_{i-1} \tilde{b}_{i-1} - \tilde{f}_{i-m+3} \tilde{d}_{i-m+3} \tilde{g}_{i-m+3} - \tilde{e}_{i-m+2} \tilde{d}_{i-m+2} \tilde{f}_{i-m+2} - \tilde{c}_{i-m+1} \tilde{d}_{i-m+1} \tilde{e}_{i-m+1} \\
\tilde{h}_i &= -\tilde{e}_{i-m+3} \tilde{d}_{i-m+3} \tilde{g}_{i-m+3} - \tilde{c}_{i-m+2} \tilde{d}_{i-m+2} \tilde{f}_{i-m+2}- \\
& & i=1,2,\dots,N \\
\tilde{g}_i &= -\tilde{e}_{i-2} \tilde{d}_{i-2} \tilde{h}_{i-2} - \tilde{f}_{i-1} \tilde{d}_{i-1} \tilde{b}_{i-1} \\
\tilde{f}_i &= -\tilde{c}_{i-2} \tilde{d}_{i-2} \tilde{h}_{i-2} - \tilde{e}_{i-1} \tilde{d}_{i-1} \tilde{b}_{i-1} \\
\tilde{e}_i &= -\tilde{c}_{i-1} \tilde{d}_{i-1} \tilde{b}_{i-1} \\
\tilde{c}_i &= c_i
\end{aligned}$$

The non-defined elements are zero.

The resulting ICCG(2,4) process necessitates about 24N multiplications per iteration and requires six arrays of length N for the decomposition.

If, instead of the two extra non-zero diagonals $\tilde{h}$ and $\tilde{g}$ in L^T, we take only the $\tilde{h}$-diagonal, then the ICCG(3)-method of Ref 1 arises. This method is denoted in this report by ICCG(2,3).

2.1.6 Some other decompositions

Proceeding in this manner, we obtain a sequence of incomplete decompositions $K_{3,6}$, $K_{5,9}$, $K_{8,14}$, $K_{13,22}$, etc. From this we see that the number of non-zero diagonals grows rapidly and thus the amount of work. However, for most practical problems the two diagonals next to those of the previous decomposition cause the major improvement. In this way K(3,5) K(4,6) etc., together with the corresponding ICCG methods ICCG(3,5), ICCG(4,6) etc., are developed.
Up to now we have always added complete diagonals in the decomposition process. The diagonals, however, contain their non-zero entries only in a blockwise structure (see Fig. 9). In particular, this implies that in our terminology a complete choleski factorisation is equivalent to "incomplete decomposition" with 2m-2 extra "diagonals".

2.2 Five-point discretisation for problems with periodic boundary conditions

The linear equations in this section arise in the same way as the linear equations in section 2.1, except that a periodic boundary condition holds in at least one direction. In the examples we have restricted ourselves to a periodic boundary condition in the x-direction. This boundary condition gives rise to additional elements in the matrix A, as indicated in Fig. 10. These extra elements are denoted by β_i. Since i is the row index, only β_1, β_{m+1}, β_{2m+1},... are non-zero. Again A is an M-matrix.

2.2.1 ICCG(1,1)

In common with the non-periodic case in 2.1.1 an incomplete decomposition can be constructed in cases where the upper triangular factor has the same sparsity structure as the upper triangular part of A. This decomposition is written as

$$K_{1,1} = L_{1,1} D_{1,1} L_{1,1}^T$$

The elements of $L_{1,1}^T$ and $D_{1,1}$ can be calculated from

$$\tilde{a}_i = \tilde{d}_i^{-1} = a_i - \tilde{b}_{i-1}^2 \tilde{d}_{i-1} - \tilde{\beta}_{i-m+1}^2 \tilde{d}_{i-m+1} - \tilde{c}_{i-m}^2 \tilde{d}_{i-m} \qquad i=1.2,\ldots N$$

$$\tilde{b}_i = b_i \qquad \tilde{c}_i = c_i \qquad \tilde{\beta}_i = \beta_i$$

Non-defined elements are zero. Note that the term $\tilde{\beta}_{i-m+1}^2 \tilde{d}_{i-m+1} \neq 0$ only if i=m, 2m, 3m,.... The resulting ICCG(1,1) process again takes approximately 16N multiplications per iteration.

2.2.2 ICCG(1,2)

The matrix $K_{1,1}$ of 2.2.1 has elements as indicated in Fig. 11. To annihilate these elements in L^T, non-zero elements are required in these places. These elements are denoted as shown in Fig. 12 by $\tilde{a}_i$, $\tilde{b}_i$, $\tilde{c}_i$, $\tilde{\beta}_i$, $\tilde{e}_i$, $\tilde{\gamma}_i$, $\tilde{\delta}_i$ and the elements of $D_{1,2}$ by $\tilde{d}_i$. They can be calculated from

$$\begin{aligned}\tilde{a}_i = \tilde{d}_i^{-1} = a_i & - \tilde{b}_{i-1}^2 d_{i-1} - \tilde{e}_{i-m+1}^2 \tilde{d}_{i-m+1} - c_{i-m}^2 d_{i-m} \\ & - \tilde{\delta}_{i-1}^2 \tilde{d}_{i-1} \qquad \text{(only for i=m+1, 2m+1, 3m+1,--)} \qquad i=1,2,3\ldots N \\ & - \tilde{\gamma}_{i-m+2}^2 \tilde{d}_{i-m+2} - \tilde{\beta}_{i-m+1}^2 \tilde{d}_{i-m+1} \text{ (only for i=m, 2m---)}\end{aligned}$$

$$\tilde{b}_i = b_i - \tilde{c}_{i-m+1} \tilde{d}_{i-m+1} \tilde{e}_{i-m+1}$$

$$\tilde{c}_i = c_i$$

$$\tilde{e}_i = -\tilde{c}_{i-1} \tilde{d}_{i-1} \tilde{b}_{i-1}$$

$$\tilde{\delta}_i = -\tilde{c}_{i-m+2} \tilde{d}_{i-m+2} \tilde{\gamma}_{i-m+2} - \tilde{c}_{i-m+1} \tilde{d}_{i-m+1} \tilde{\beta}_{i-m+1} \qquad i=m,\ 2m---$$

$$\tilde{\gamma}_i = -\tilde{\beta}_{i-1} \tilde{d}_{i-1} \tilde{b}_{i-1} \qquad i=2,\ m+2,\ 2m+2,---$$

$$\tilde{\beta}_i = \beta_i - \tilde{c}_{i-1} \tilde{d}_{i-1} \tilde{\delta}_{i-1} \qquad i=1,\ m+1,\ 2m+1.---$$

Note that $\tilde{b}_m$, $\tilde{b}_{2m}$,-- and $\tilde{e}_1$, $\tilde{e}_{m+1}$,--- are non-zero.
The resulting ICCG process takes 18N multiplications per iteration and requires roughly three arrays of length N for the incomplete decomposition.

2.2.3 ICCG(1,1) periodic

The incomplete decomposition of 2.2.1 does not have a periodic structure. To obtain a K with a periodic structure we write

$$K_p = (L_p + D_p{}^{-1})\, D_p\, (L_p^T + D_p^{-1})$$

The periodic structure of the matrix L_p is given in Fig. 13. D_p is a diagonal matrix. The elements of L_p and D_p have to satisfy:

$$\tilde{b}_i = b_i \qquad \tilde{c}_i = c_i \qquad \tilde{\beta}_i = \beta_i \qquad \text{for } i=1,\ 2,\text{--}N \tag{2.2.3.1}$$

$$\tilde{a}_i = \tilde{d}_i^{-1} = a_i - \tilde{b}_{i-1}^{\,2}\, \tilde{d}_{i-1} - \tilde{c}_{i-m}^{\,2}\, \tilde{d}_{i-m} \qquad \text{for } i=2,\ 3,\text{--},\ m,\ m+2,\ m+3,\text{--},2m,2m+2,\ \ldots,N \tag{2.2.3.2}$$

$$\tilde{a}_i = \tilde{d}_i^{-1} = a_i - \tilde{\beta}_i^2\, \tilde{d}_{i+m-1} - \tilde{c}_{i-m}^{\,2}\, \tilde{d}_{i-m} \qquad \text{for } i=1,\ m+1,\ 2m+1,\text{--},\ N-m+1 \tag{2.2.3.3}$$

The $\tilde{d}_i$ cannot be calculated straightforwardly, since in the second formula $\tilde{d}_{i+m-1}$ is present. We can calculate them by substituting (2.2.3.3) into (2.2.3.2) for the next i, and continuing in this way, we find quadratic equations for the $\tilde{d}_{km}$. For $\tilde{d}_{km}$ we choose the largest root, since this choice results in smaller elements $\tilde{b}_i\, \tilde{d}_{i-1}^{-1}\, \tilde{c}_{i-m-1}$ in the error matrix K_p-A. We now give the derivation for the formula for $\tilde{d}_m$. The $\tilde{d}_{km}$ can be computed in a similar way. We rewrite (2.2.3.3) as

$$\tilde{d}_1^{-1} = w_1 - v_i\, \tilde{d}_m \tag{2.2.3.4}$$

and 2.2.3.2 as

$$\tilde{d}_i^{-1} = w_i - v_i\, \tilde{d}_{i-1} \qquad i=2,\ \text{---}m \tag{2.2.3.5}$$

From induction it follows that:

$$\tilde{d}_i = \frac{p_i + q_i\, \tilde{d}_m}{r_i + s_i\, \tilde{d}_m} \qquad \text{for } i=1,\ 2,\ \text{---},m$$

The coefficients p_i, q_i and s_i satisfy $p_1 = 1 \quad q_1 = 0 \quad r_1 = w_1 \quad s_1 = -v_1$
$p_{i+1} = r_i \quad q_{i+1} = s_i \quad r_{i+1} = w_{i+1}\, r_i - \tilde{v}_{i+1}\, p_i \quad s_{i+1} = w_{i+1}\, s_i -- v_{i+1} q_i$
This leads to the quadratic equation in $\tilde{d}_m$ with known coeeficients p_m, q_m, r_m and s_m:

$$s_m \tilde{d}_m^2 + (r_m - q_m) \tilde{d}_m - p_m = 0$$

from which the largest root can be calculated.

2.3 Seven-point discretisations of elliptic p.d.e.'s in three dimensions

The seven-point discretisation for equation (2.1.1) in three dimensions leads in a similar way to a matrix with seven non-zero diagonals. The structure of this matrix A is shown in Fig. 14. The elements of the upper triangular part of A are denoted by a_i, b_i, c_i and e_i, where i is counted rowwise. If n, m, k are the number of gridpoints in the x,y,z directions, respectively, the order of the matrix and the sizes of the blocks are n x m x k, n x m and n.

2.3.1 ICCG(1,1,1)

In common with the 2-D case the ICCG (1,1,1) factorisation is the one where the upper triangular factor has the same non-zero structure as the upper triangular part of A. Again this decomposition is written as $K_{1,1,1} = L_{1,1,1} D_{1,1,1} L^T_{1,1,1}$, where $L^T_{1,1,1}$ is an upper triangular matrix and $D_{1,1,1}$ a diagonal matrix equal to the inverse of the main diagonal of $L^T_{1,1,1}$. The elements of $L^T_{1,1,1}$ are denoted by $\tilde{a}_i$, $\tilde{b}_i$, $\tilde{c}_i$ and $\tilde{e}_i$ and the elements of $D_{1,1,1}$ by $\tilde{d}_i$. These elements are given by the recurrency relations:

$$\tilde{a}_i = \tilde{d}_i^{-1} = a_i - \tilde{b}_{i-1}^2 \tilde{d}_{i-1} - \tilde{c}_{i-n}^2 \tilde{d}_{i-n} - \tilde{e}_{i-mn}^2 \tilde{d}_{i-mn} \qquad \text{for } i=1,2,---,n \times m \times k$$

$$\tilde{b}_i = b_i \qquad \tilde{c}_i = c_i \qquad \tilde{e}_i = e_i$$

Non-defined elements should be replaced by zeros. It can easily be seen that for major problems where the diagonals cannot be stored all together in core, the $\tilde{d}_i$ can be calculated by taking successively only parts of the order of n x m in core.

The resulting hybrid conjugate gradient method requires 20 N multiplications per iteration.

2.3.2 Other decompositions for 3-D

The matrix $K_{1,1,1} = L_{1,1,1}D_{1,1,1}L^T_{1,1,1}$, of the previous section is a matrix equal to A, except for six diagonals, as shown in Fig. 15 as dotted lines. We obtain the next decomposition by including non-zero entries on these lines in L and L^T. The elements of L^T are denoted by $\tilde{a}_i$, $\tilde{b}_i$, $\tilde{c}_i$, $\tilde{e}_i$, $\tilde{f}_i$, $\tilde{g}_i$ and $\tilde{h}_i$, as shown in Fig. 16, and can be calculated from:

$$\tilde{a}_i = \tilde{d}_i^{-1} = a_i - \tilde{b}^2_{i-1}\,\tilde{d}_{i-1} - \tilde{h}^2_{i-n+1}\,\tilde{d}_{i-n+1} - \tilde{c}^2_{i-n}\,\tilde{d}_{i-n} - \tilde{g}^2_{i-nm+n}\,\tilde{d}_{i-nm+n}$$

$$- \tilde{f}^2_{i-nm+1}\,\tilde{d}_{i-nm+1} - \tilde{e}^2_{i-nm}\,\tilde{d}_{i-nm}$$

$$\tilde{b}_i = b_i - \tilde{c}_{i-n+1}\,\tilde{d}_{i-n+1}\,\tilde{h}_{i-n+1} - \tilde{e}_{i-mn+1}\,\tilde{d}_{i-mn+1}\,\tilde{f}_{1-mn+1}$$

$$\tilde{h}_i = \tilde{c}_{i-1}\,\tilde{d}_{i-1}\,\tilde{b}_{i-1} - \tilde{f}_{i-mn+n}\,\tilde{d}_{i-mn+n}\,\tilde{g}_{i-mn+n}$$

$$i=1,\dots,nmk$$

$$\tilde{c}_i = c_i - \tilde{e}_{i-mn+n}\,\tilde{d}_{i-mn+n}\,\tilde{g}_{i-mn+n}$$

$$\tilde{g}_i = -\tilde{f}_{i-n+1}\,\tilde{d}_{i-n+1}\,\tilde{h}_{i-n+1} - \tilde{e}_{i-n}\,\tilde{d}_{i-n}\,\tilde{c}_{i-n}$$

$$\tilde{f}_i = -\tilde{e}_{i-1}\,\tilde{d}_{i-1}\,\tilde{b}_{i-1}$$

$$\tilde{e}_i = e_i$$

Six arrays of length N are required to store the non-zero diagonals of L^T. The resulting ICCG method requires 26N multiplications per iteration. Unfortunately, if we proceed in this manner the number of diagonals in the subsequent decompositions will increase rapidly. For instance, the next decomposition has 12 non-zero diagonals in its upper triangular part. The resulting ICCG method takes 36 N multiplications per iteration.

2.4 M-matrices arising from five-point discretisations on irregular regions

So far we have only considered discretisations on square regions. We are now going to comment on regions with internal boundaries (no-flow boundaries) or differently shaped regions. For convenience it will be assumed that the region consists of small squares.

An internal boundary is reflected by some extra zeros in the matrix, but the matrix remains a symmetric M-matrix, thus incomplete decompositions can be constructed as before. An internal boundary implies that there is no direct connection (no flow) between points on different sides of the boundary. This property is preserved in each of the above-mentioned decompositions. This is in contrast with Stone's SIP method[2], where the use of the iteration parameter may cause a connection through a no-flow boundary. Irregularly shaped regions can be extended in an obvious way to square regions with an internal boundary at the point of the original real boundary. The linear system arising from this extended region can be treated as before, bearing in mind that the extended parts do not require computational work.

2.5 M.matrices with an irregular non-zero structure

M-matrices with an irregular non-zero structure arise, for instance, from some finite-element methods on irregular meshes[11] and pipeline networks[12].

We write the matrix A as $A = L + D + U$ where L, U are strictly lower and upper triangular, respectively, and D the diagonal of A. If we omit all Gaussian elimination corrections on off-diagonal elements (see section 1), then the incomplete decomposition is given by $K_o = (L + D_o)\, D_o^{-1}\, (D_o + U)$. D_o is determined by the relation that the diagonal of K_o-A, which is equal to the diagonal of $D_o + \mathrm{diag}(LD_o^{-1}U) - D$, is zero.

If the matrix is symmetric, this decomposition can be combined with the conjugate gradient method. For non-symmetric matrices see section 4.

3. ALGORITHMS FOR SYMMETRIC POSITIVE DEFINITE MATRICES

If the matrix is not an M-matrix, the construction of an incomplete decomposition may fail, because of the occurrence of non-positive diagonal elements[5]. Small positive diagonal elements are also undesirable, because of stability problems.

Three different strategies which seem to overcome this problem have currently been proposed:

(i) If a diagonal element of less than a prescribed positive value is encountered during the construction of the incomplete LL^T decomposition then some already computed off-diagonal elements in the corresponding column of L^T are set to zero.

(ii) The diagonal element can be enlarged if necessary by adding a sufficient amount to the original element[5].

(iii) We can also add αI to the matrix[6]. If α is large enough, the signalled problems will not occur.

4. ALGORITHMS FOR NON SYMMETRIC POSITIVE DEFINITE MATRICES

If the matrix is non-symmetric, then an incomplete LU decomposition K can be constructed in a similar way as described previously for the symmetric matrices. Since symmetry and positive-definiteness are both required for the conjugate gradient algorithm, the CG algorithm can be applied:

$$A^T K^{T^{-1}} K^{-1} A\, x = A^T K^{T^{-1}} K^{-1} b$$

This algorithm requires twice as much work per iteration as the corresponding symmetric case and the upper bound for the number of iterations increases by a factor of two.

5. NUMERICAL EXPERIMENTS

To obtain an impression of the convergence behaviour of different incomplete decompositions, we have for the ICCG-methods introduced in section 2.1:

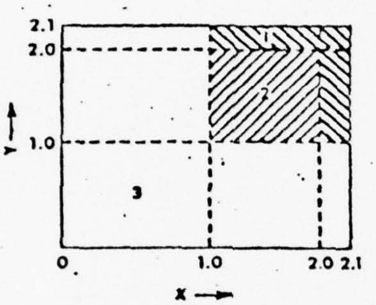
(i) compared the convergence results

(ii) calculated the eigenvalue distribution of the preconditioned matrices $K^{-1}A$.

The two test problems were:

i) problem 1. The five-point discretisation of the Poisson equation $\Delta u=0$ over $0 \leq x \leq 1$, $0 \leq y \leq 1$ with boundary conditions $\frac{\partial u}{\partial x} = 0$ for $x = 0$ and $x = 1$, $\frac{\partial u}{\partial y} = 0$ for $y = 1$ and $u = 1$ for $y = 0$. A uniform rectangular mesh was chosen, with $\Delta x = 1/31$ and $\Delta y = 1/31$, which resulted in a linear system of 992 equations. The solution of this equation is known to be $u(x,y) = 1$ and as initial starting vector for the iterative schemes a vector was chosen with all entries random between 0 and 1. This was done to prevent coincidental fast convergence.

ii) problem 2. This problem has been taken from Varga[7]. Equation (2.1.1) holds for R, where R is the square region $0 \leq x, y \leq 2.1$ as shown below



On the boundary of R, the boundary conditions are $\frac{\partial u}{\partial n} = 0$. Further, $D(x,y) = 0$ over R and the functions A, B and C are given by

Region	A(x,y)	B(x,y)	C(x,y)
1	1.0	1.0	0.02
2	2.0	2.0	0.03
3	3.0	3.0	0.05

A uniform rectangular mesh was chosen with 0.05 mesh spacing, so that a system of 1849 linear equations resulted. The solution of the system is known to be $u = 0$. A vector similar to the one in problem I was chosen as a starting vector.

In Tables 1 and 2 the convergence results are listed. In both tables we can conclude from the last column that ICCG(1,3) is almost optimal. Since the convergence behaviour depends on the eigenvalue distribution, where the condition number and clustering play an important role, a number of the lowest and highest eigenvalues have been calculated.

The eigenvalues are all divided by λ_{min} for the matrix of problem 1, preconditioned with several incomplete decompositions because an upper bound for the convergence of the conjugate gradient method is given by $(\sqrt{c}-1)/(\sqrt{c}+1)$, where $c = \lambda_{max}/\lambda_{min}$. The distribution of these scaled eigenvalues has been plotted in Figs. 17-21.

REFERENCES

1. Meyerink, J.A. & van der Vorst, H.A., An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix.
   Math. of Comp., Vol. 31, No. 137, 1977, pp. 148-162.

2. Stone, H.L., Iterative solution of implicit approximations of multidimensional partial differential equations.
   SIAM J. Numer.Anal., Vol.5, 1968, pp. 530-558

3. Dupont, T., Kendall, M.R.P. & Rachford, H.H., An approximate factorisation procedure for solving self-adjoint elliptic difference equations.
   SIAM J. Numer. Anal., Vol. 5, 1968, pp. 559-573.

4. Gustafsson, I., A class of 1:st order factorisation methods.
   Research Report, Dept. of Comp. Sciences, University of Göteborg.

5. Kershaw, D.S., The incomplete Choleski-conjugate gradient method for the iterative solution of systems of linear equations.
   J. of Comp. Physics, Vol. 26(1), 1978, pp. 43-65.

6. Manteuffel, T.A., Incomplete Choleski decomposition.
   Very informal proceedings of the Gatlinburg VII meeting on numerical linear Algebra, 1978.

7. Varga, R.S., Matrix Iterative Analysis, Prentice-Hall, Englewood Cliffs N.J., 1962.

8. van der Sluis, A., Condition numbers and equilibration of matrices.
   Numer. Math., Vol. 14, 1969, pp. 14-23.

9. Varga, R.S., Factorisation and normalised iterative methods.
   Boundary Problems in Diff. Eq. (R.E. Langer, ed.), Univ. of Wisconsin Press, 1960, pp. 121-142.

10. Axelsson, O., A generalised SSOR method.
    BIT Vol. 13, 1972, pp. 443-467.

11. Strang, G. & Fix, G.J., An analysis of the finite element method.
    Prentice-Hall, Englewood Cliffs N.J., 1973.

12. Porsching, T.A., On the Origins and Numerical Solution of Some Sparse Nonlinear Systems.
    Sparse Matrix Computations, Academic Press, New York, 1976.

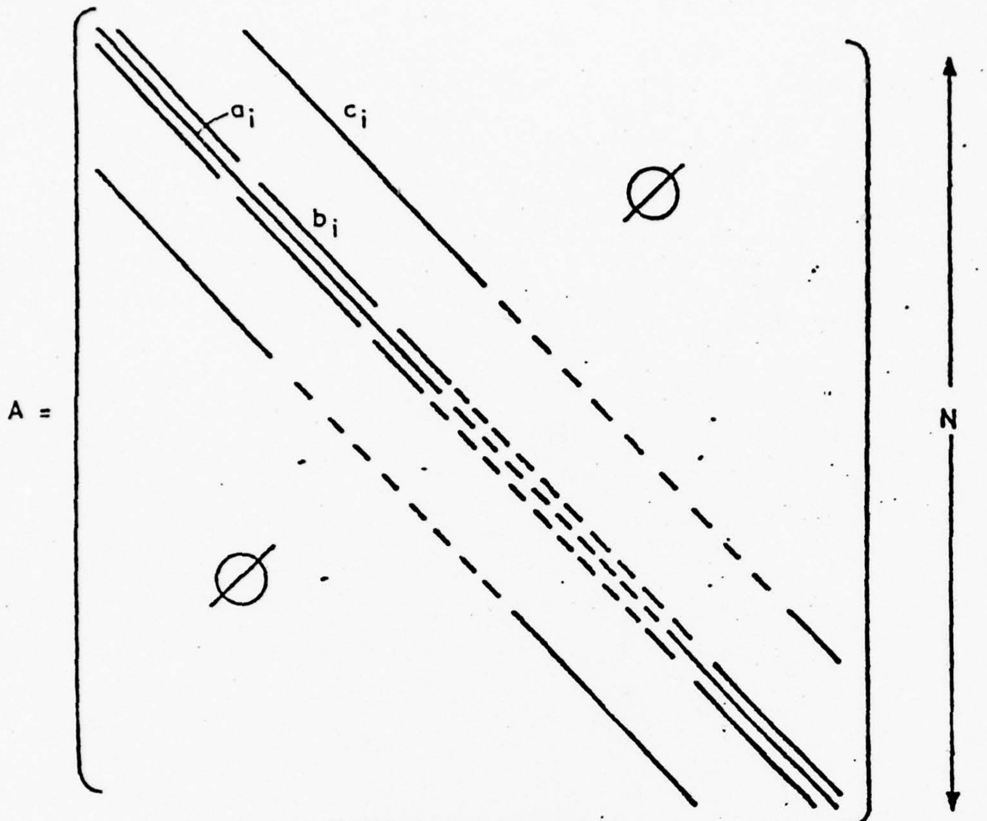


FIG. 1

(TWO DIMENSIONAL)
MATRIX OF SECTION 1.

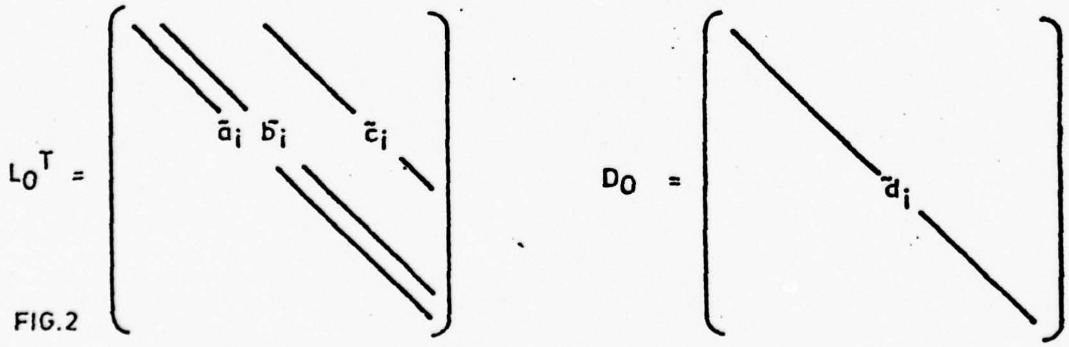


FIG. 2

MATRICES $L^T_{1,1}$ AND $D_{1,1}$ (SECTION 2.1.2)

FIG.1.2

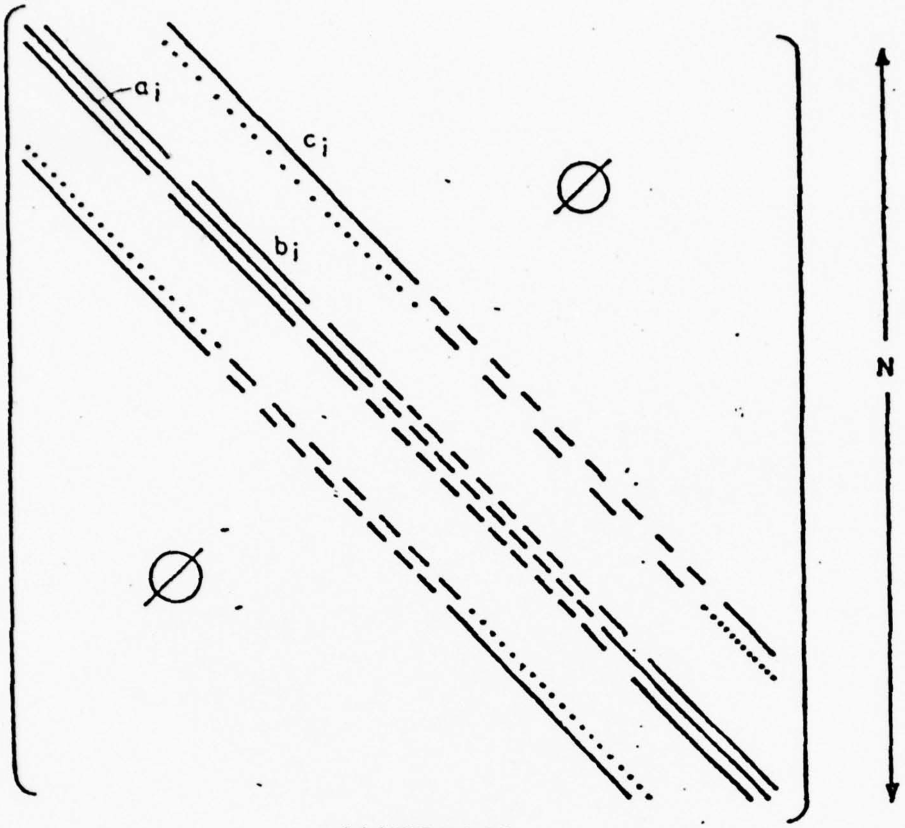


FIG.3

MATRIX $K_{1,1}$

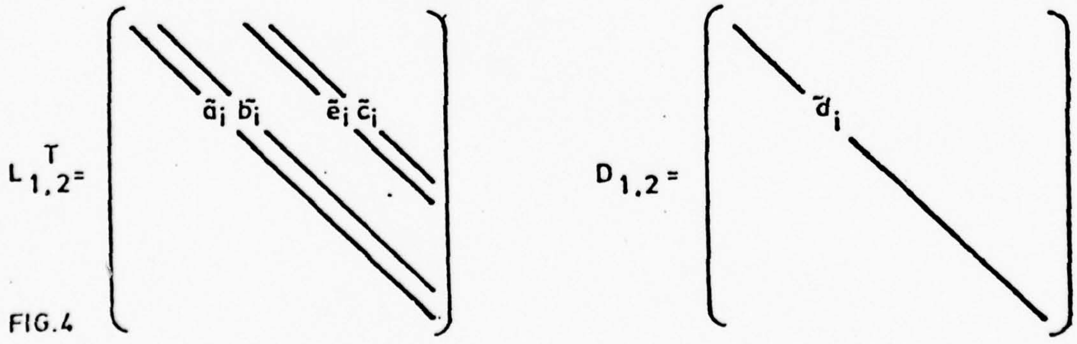


FIG.4

MATRICES $L^{T}_{1,2}$ AND $D_{1,2}$ (SECTION 2.1.3)

FIG. 3.4

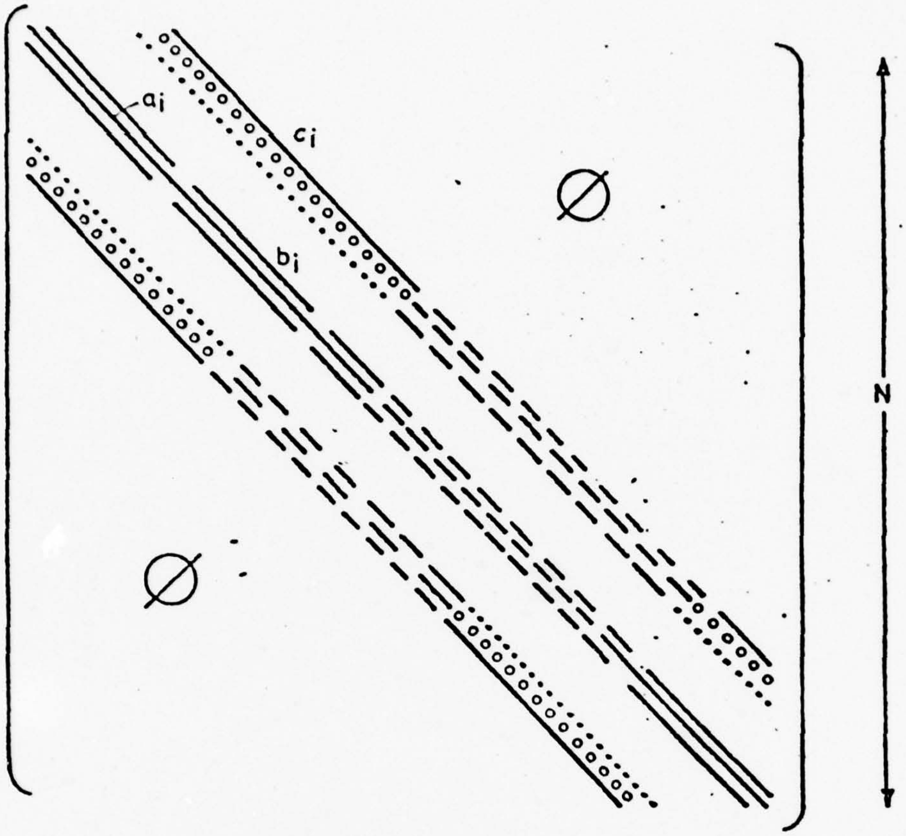


FIG.5

MATRIX $K_{1,2}$

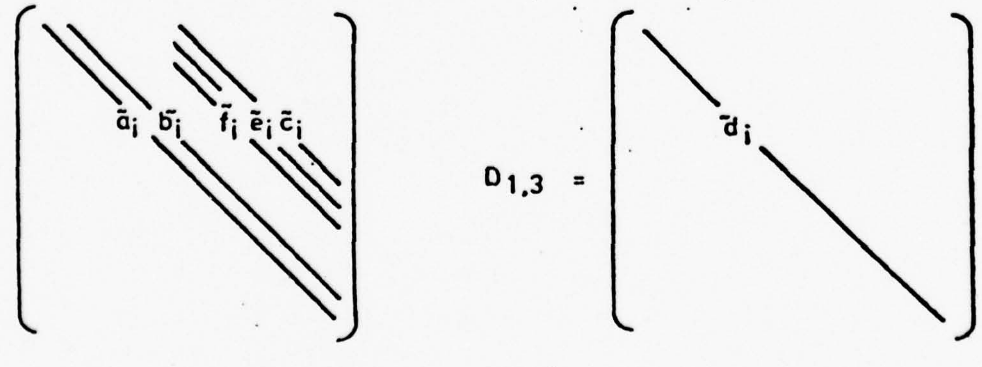


FIG.6

MATRICES $L^T_{1,3}$ AND $D_{1,3}$ (SECTION 2.1.4)

FIG.5,6

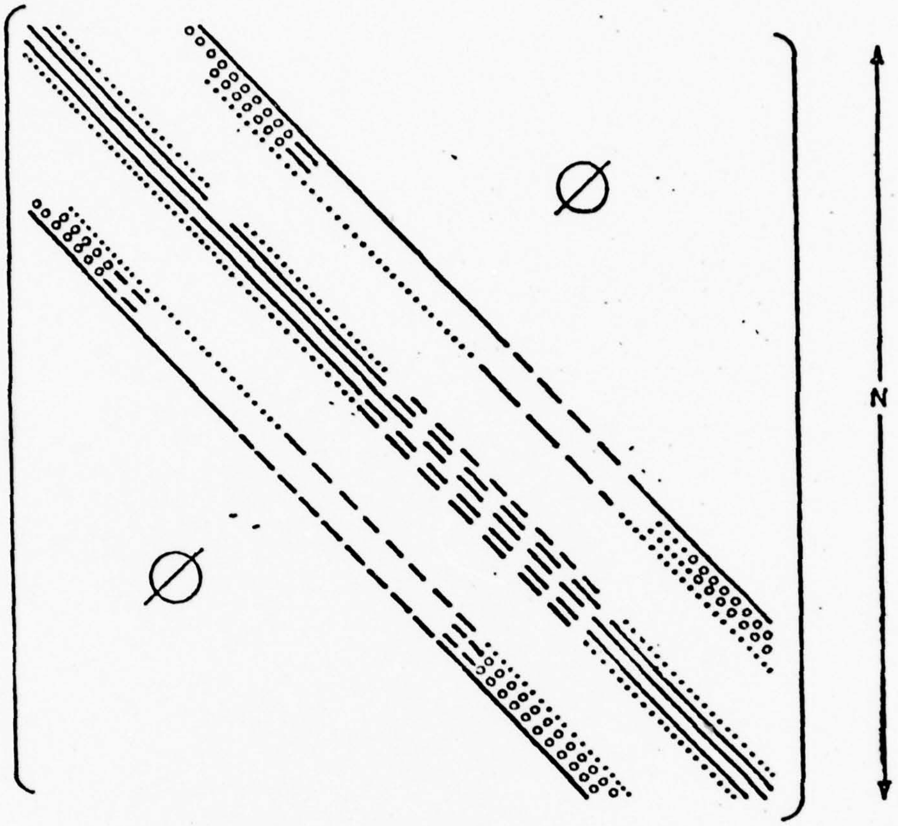


FIG. 7

MATRIX $K_{1,3}$ (SECTION 2.1.5)

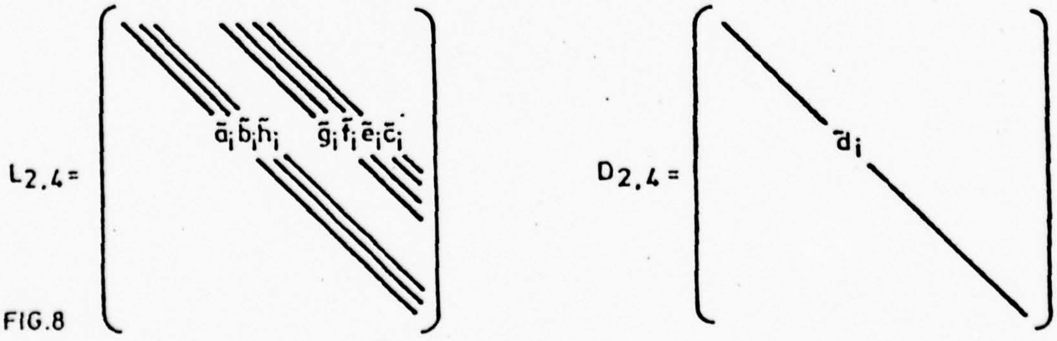
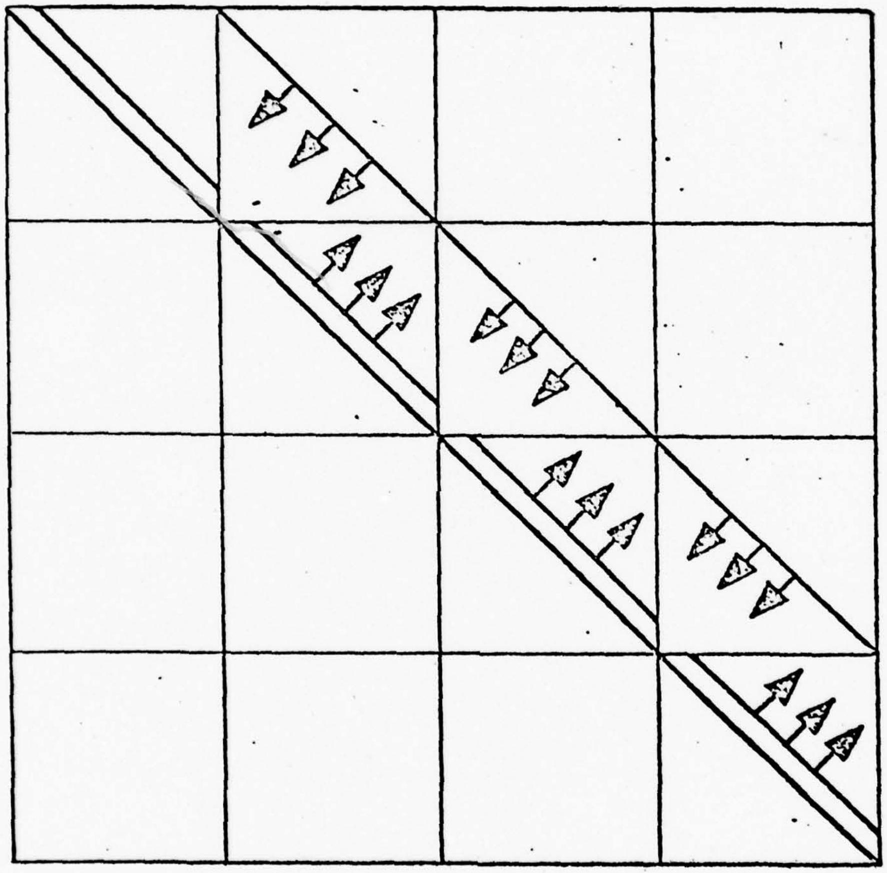

$L_{2,4}=$
$\bar{a}_i \bar{b}_i \bar{h}_i$
$\bar{g}_i \bar{f}_i \bar{e}_i \bar{c}_i$
$D_{2,4}=$
$\bar{d}_i$

FIG.8

MATRICES $L_{2,4}$ AND $D_{2,4}$

FIG.7,8



STRUCTURE OF CHOLESKI FACTORISATION

FIG. 9

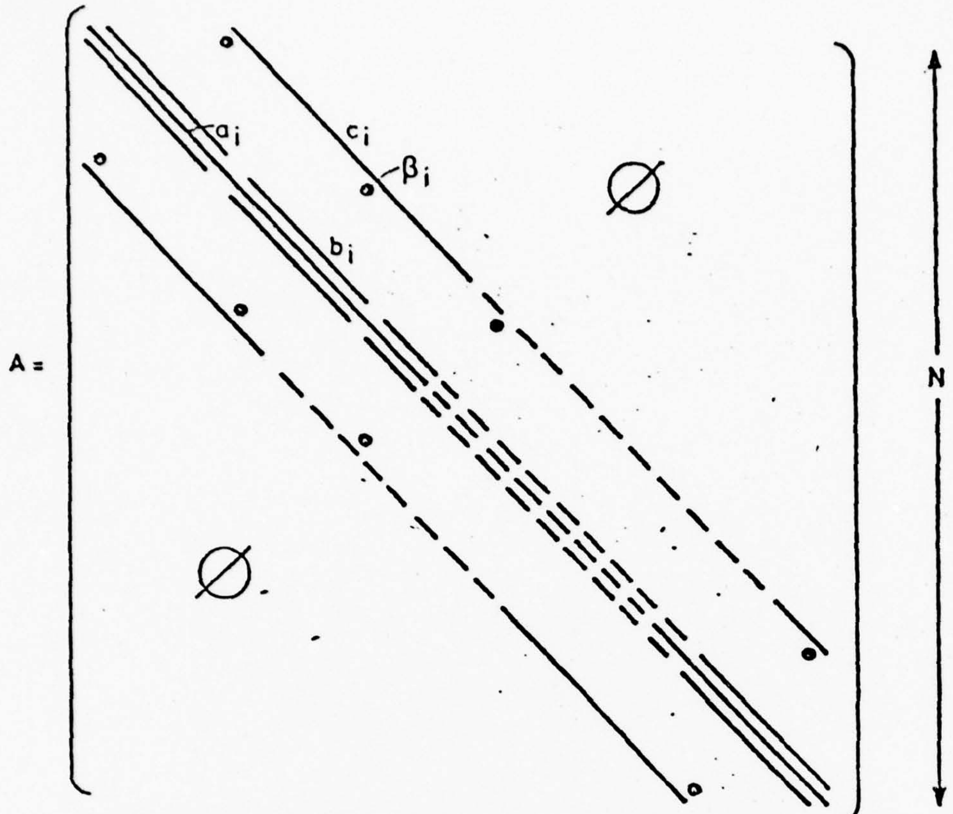


FIG. 10

MATRIX A PERIODIC BOUNDARY CONDITIONS

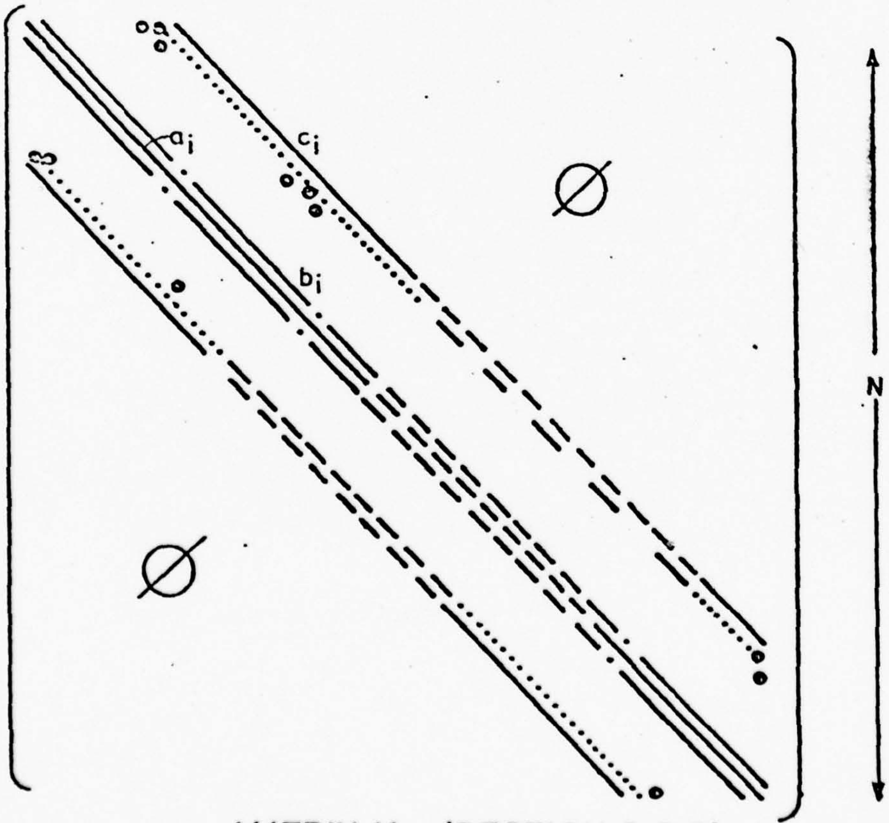


FIG.11

MATRIX $K_{1,1}$ (SECTION 2.2.2)

FIG. 10,11

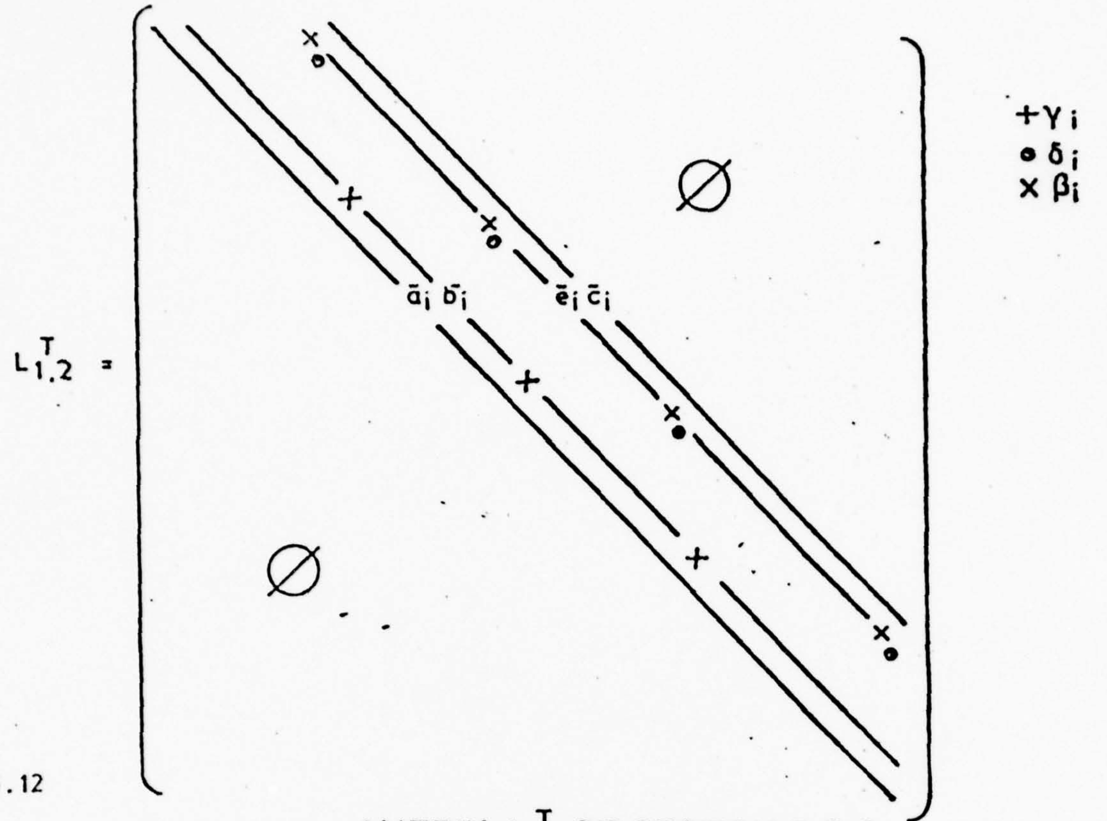


FIG. 12

MATRIX $L^T_{1,2}$OF SECTION 2.2.2

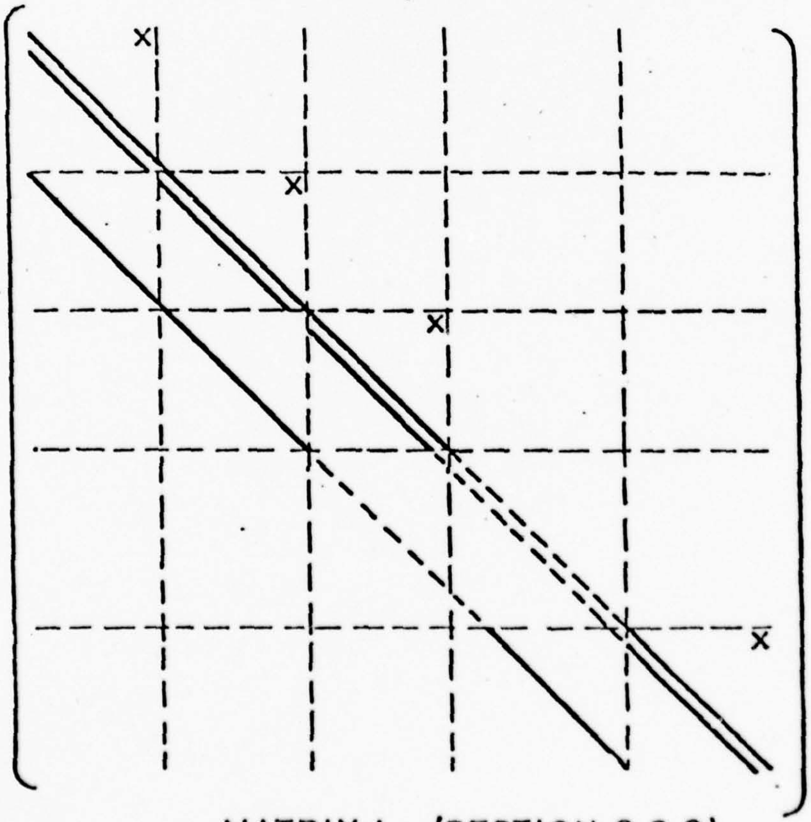


FIG.13

MATRIX L_p (SECTION 2.2.3)

FIG.12 ,13

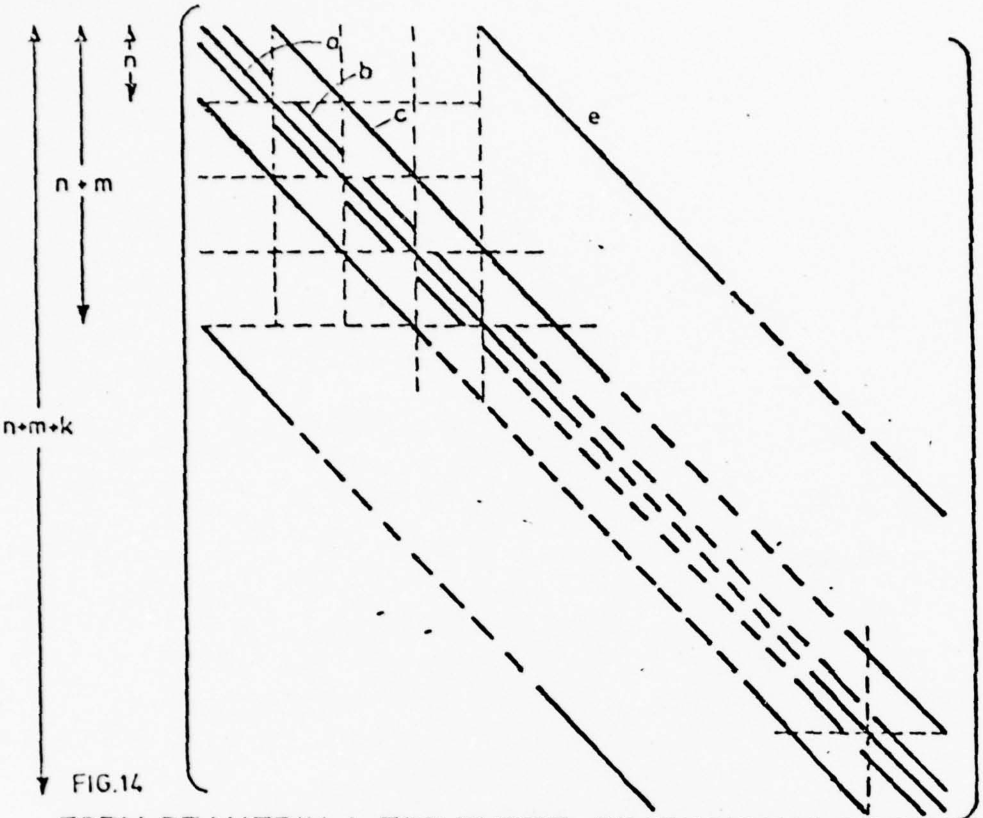


FIG.14

FORM OF MATRIX-A FOR THREE-DIMENSIONAL PROBLEMS

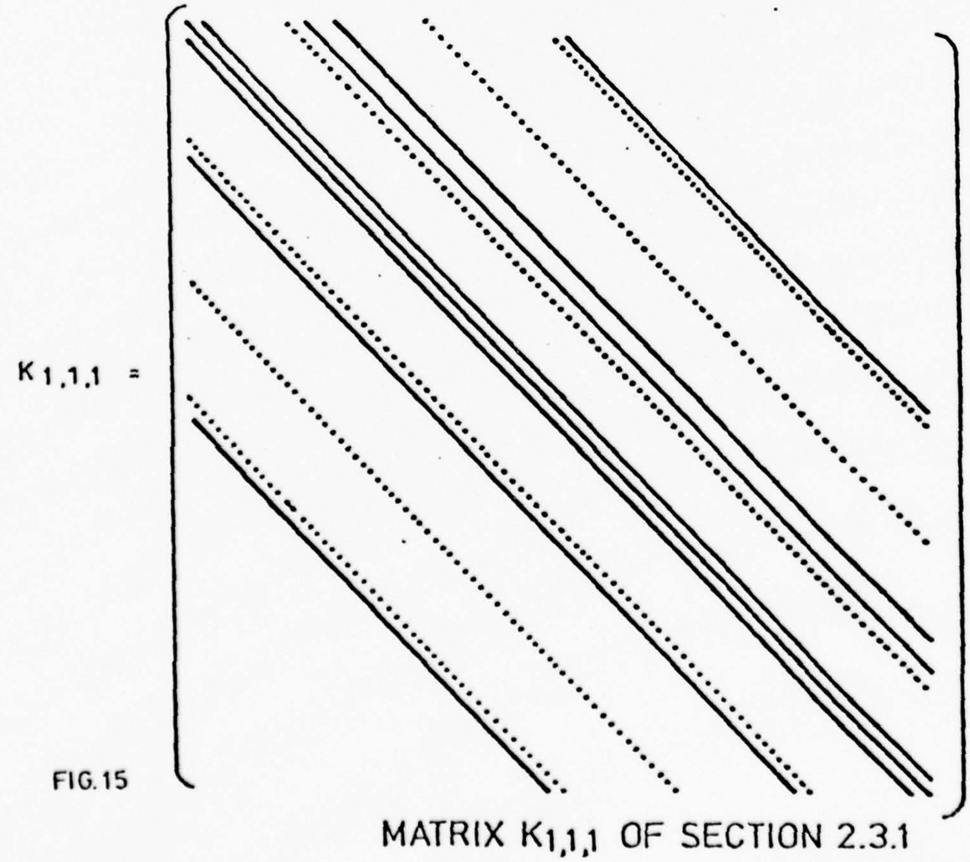


FIG.15

MATRIX $K_{1,1,1}$ OF SECTION 2.3.1

FIG.14,15

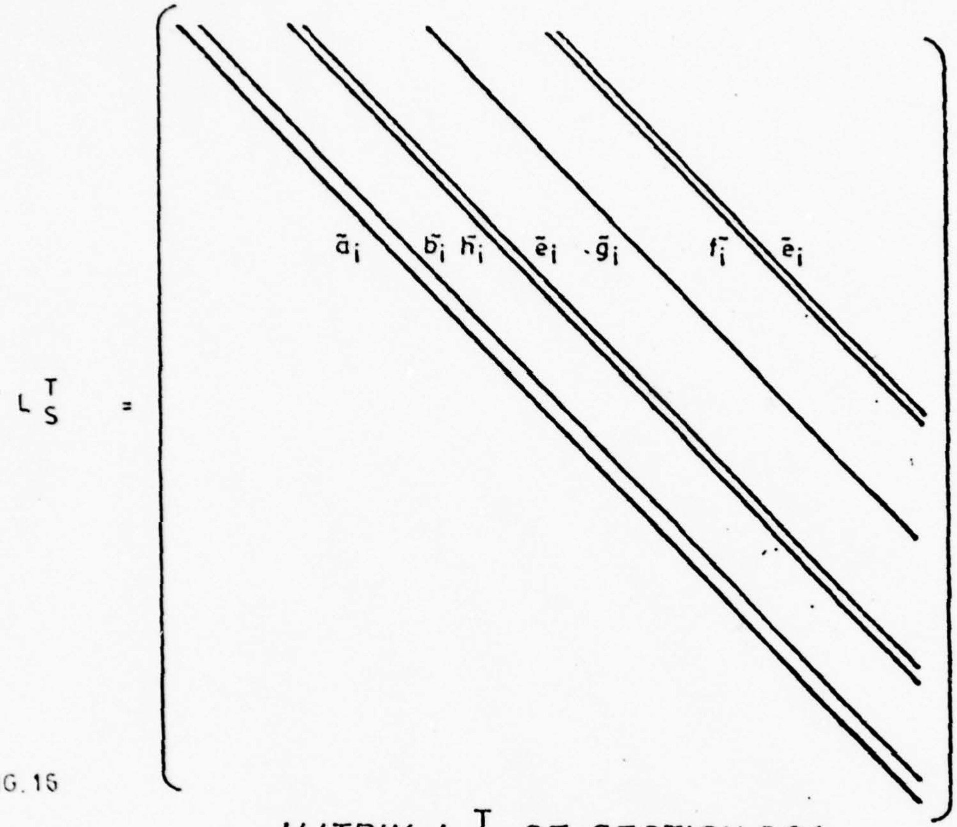


FIG. 16

MATRIX L^T_3 OF SECTION 2.3.1

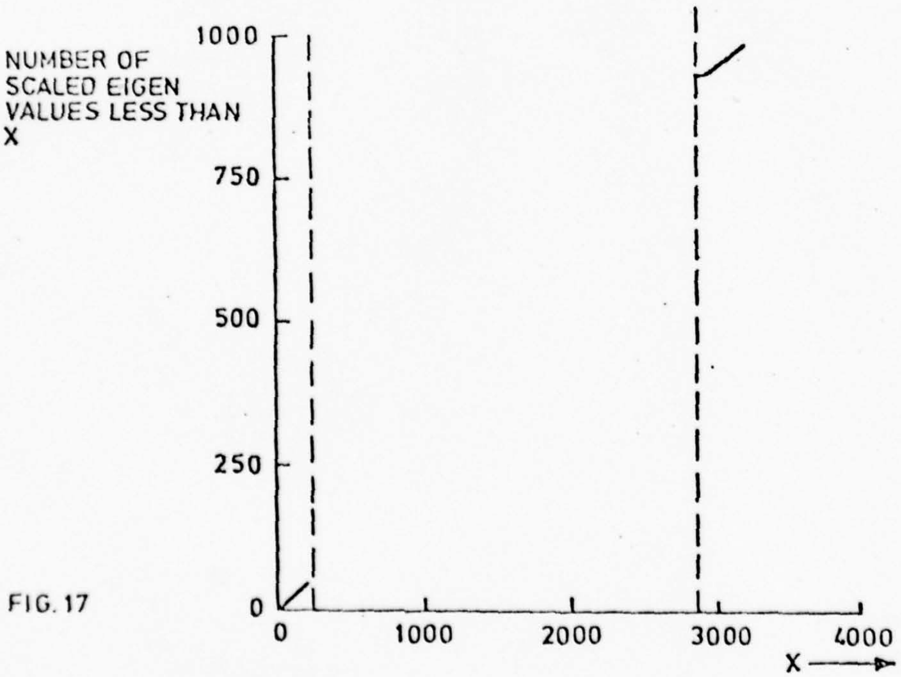


FIG. 17

DISTRIBUTION OF SCALED EIGENVALUES OF A

FIG. 16.17

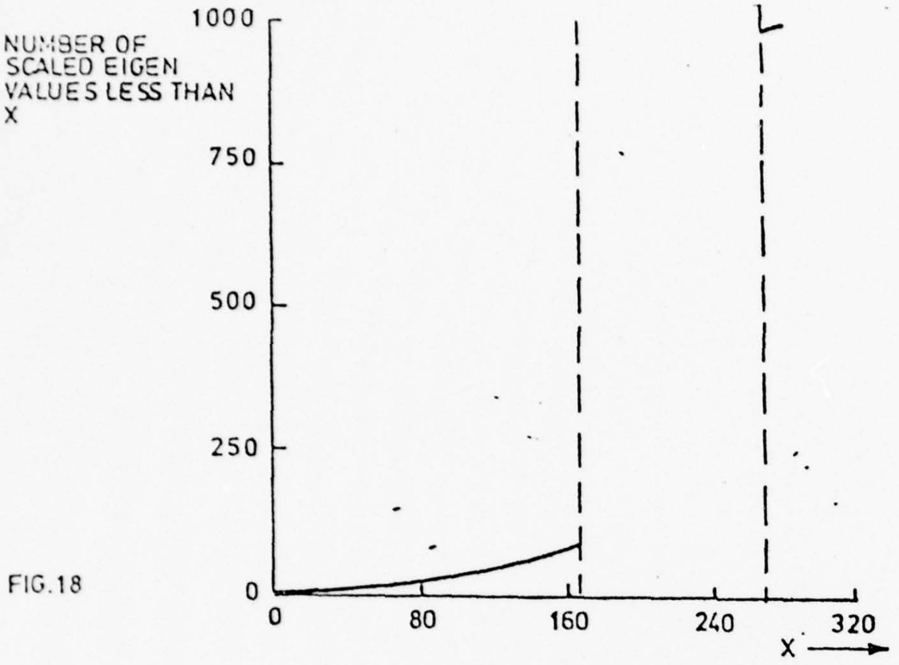


FIG.18

DISTRIBUTION OF SCALED EIGENVALUES OF $K_{1,2}^{-1} A$

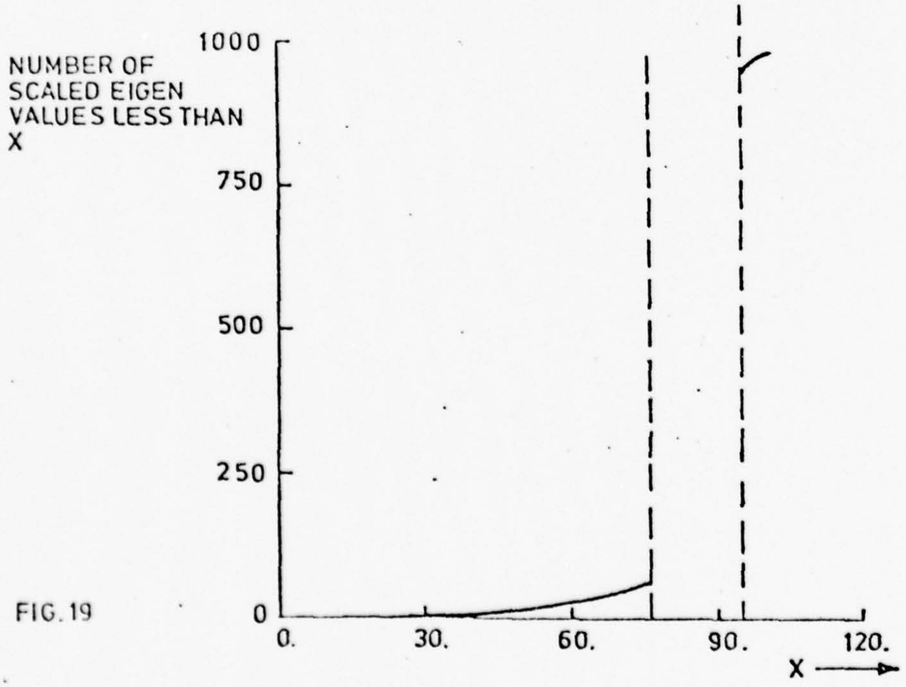


FIG.19

DISTRIBUTION OF SCALED EIGENVALUES OF $K_{1,2}^{-1} A$

FIG.18.19

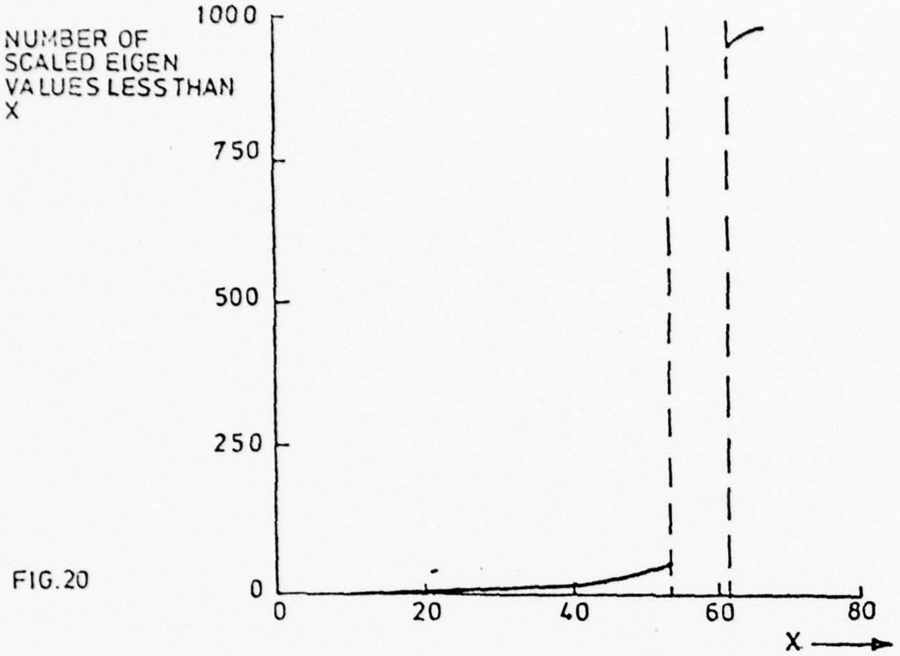


FIG.20

DISTRIBUTION OF SCALED EIGENVALUES OF $K_{1,3}^{-1}A$

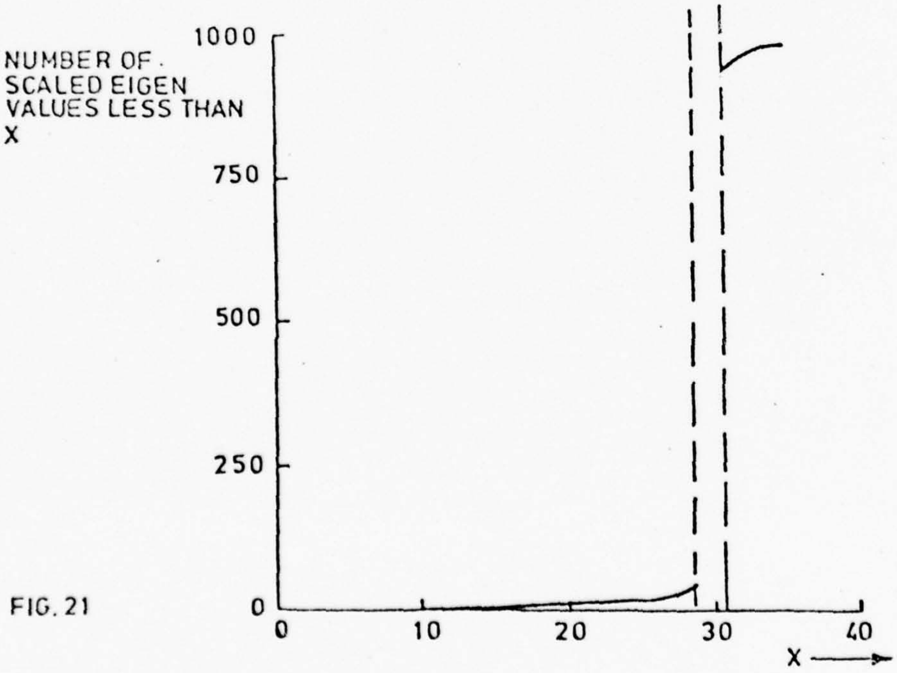


FIG. 21

DISTRIBUTION OF SCALED EIGENVALUES OF $K_{2,4}^{-1}A$

FIG.20.21.

APPENDIX B

Iterative methods for a class of nonsymmetric systems of linear equations, based on splitting-off a symmetric part.

by

H.A. van der Vorst.

Iterative methods for a class of non-symmetric systems of linear equations, based on splitting-off a symmetric part.

Henk A. van der Vorst

1. Introduction.

In recent papers Concus & Golub /1/ and Widlund /2/ discuss conjugate gradient like iterative methods for the iterative solution of real non-symmetric algebraic equations.

$$A\ x = b \qquad (1.1)$$

These methods are based on the splitting of the matrix A in a symmetric and a skew-symmetric part

$$A = \tfrac{1}{2}(A + A^T) + \tfrac{1}{2}(A - A^T) \qquad (1.2)$$

and the requirements are that $\frac{1}{2}(A + A^T)$ is positive definite. In this paper a class of splittings of the matrix A is considered and the influence of the special choice of these splittings on the convergence of a simple related iterative method is discussed.
A combination of these splittings and the incomplete choleski factorization, described by Meijerink & van der Vorst /4/, has been treated in more detail. Simple numerical experiments, showing the various effects are demonstrated. Methods for acceleration, such as those based on Manteuffel's ideas /3/, are not considered here, although they might be very effective, since the eigenvalues of the occurring iteration-matrices are located in the complex plane on straight lines parallel to the Y-axis.

2. Splitting-off a symmetric part.

Throughout this paper we think specifically of matrices that arise in the five-point discretisation of elliptic partial differential equations, that have essential, i.e. not easily removable, first order differential terms. Let us for simplicity think of the equation

$$\Delta u + \beta u_x = 0 \tag{2.1}$$

(in this case however, the first-order term is easily removed, if β is some continuous function).
For the matrices A, arising in this way, it follows that $A + A^T$ is a symmetric M-matrix, and thus positive definite (see Varga /5/).
Very simple splittings that take advantage of this are

splitting I $$A = \tfrac{1}{2}(A + A^T) + \tfrac{1}{2}(A - A^T) \tag{2.2}$$

and

splitting II $$A = (A + A^T) - A^T \tag{2.3}$$

Let us now consider a simple basic iterative method, that arises from the splitting $A = M - N$:

$$M x_{i+1} = b + N x_i \tag{2.4}$$

This leads for splitting I and II resp. to

$$\tfrac{1}{2}(A+A^T)\bar{x}_{i+1} = b - \tfrac{1}{2}(A-A^T)\bar{x}_i \tag{2.5}$$

and

$$(A+A^T)\bar{\bar{x}}_{i+1} = b + A^T\bar{\bar{x}}_i \tag{2.6}$$

for the solution of (1.1).
If the respective errorvectors $\bar{h}_i$ and $\bar{\bar{h}}_i$ are defined by

$$\bar{h}_i = \bar{x}_i - x \cdot \qquad \text{and} \qquad \bar{\bar{h}}_i = \bar{\bar{x}}_i - x \tag{2.7}$$

where x is the solution of Ax=b, then we have the relations

$$\bar{h}_i = -(A+A^T)^{-1}(A-A^T)\bar{h}_{i-1} \tag{2.8}$$

and

$$\bar{\bar{h}}_i = \left[\tfrac{1}{2}I - \tfrac{1}{2}(A+A^T)^{-1}(A-A^T)\right]\bar{\bar{h}}_{i-1} \tag{2.9}$$

The expressions (2.8) and (2.9) lead to a first observation. If $A - A^T$ is small, we have that splitting I results in a rather fast converging method, whereas in splitting II the convergence behaviour is limited by a factor $\frac{1}{2}$. In this case one might expect splitting I to be the most efficient one. The assumption $A - A^T$ is small, is not unreasonable, since from (2.1) the matrix $A - A^T$ arises from the first order term, and thus we have $A - A^T = O(h)$ compared to $A + A^T$.

One might however put the question what happens if $A - A^T$ is not very small ?

From matrix theory it is known that all the eigenvalues λ_j of $(A+A^T)^{-1}(A-A^T)$ are purely imaginary and let us define λ_{max} as the maximum absolute value of these eigenvalues. Then for splitting I, λ_{max} is a measure ~~for~~ of the speed of convergence, while for splitting II the convergence factor behaves like $|\frac{1}{2}+\frac{1}{2}i\lambda_{max}|$. Thus we conclude that for $1 < \lambda_{max} < \sqrt{3}$, splitting I leads to a divergent process, while splitting II still yields a converging process.

More explicitly, it follows that for $\frac{1}{3}\sqrt{3} < \lambda_{max} < \sqrt{3}$, splitting II will be the faster one. Therefore the question arises whether the convergence can be influenced by other splittings of the matrix A.

Consider now the following class of splittings

$$A = \alpha(A+A^T) + ((1-\alpha)A - \alpha A^T) \qquad (2.10)$$

where $\alpha \neq 0$ is an arbitrary real constant. The splitting (2.10) results in the iterative method

$$\alpha(A+A^T)x_{i+1} = b - [(1-\alpha)A - \alpha A^T]x_i \qquad (2.11)$$

And, for $h_i = x_i - x$, we have

$$h_i = [I - \frac{1}{2\alpha}I - \frac{1}{2\alpha}(A+A^T)^{-1}(A-A^T)]h_{i-1} \qquad (2.12)$$

The eigenvalues γ_j of the matrix

$$(1-\frac{1}{2}\alpha)I - \frac{1}{2\alpha}(A+A^T)^{-1}(A-A^T)$$

are related to the purely imaginary eigenvalues λ_j of $(A+A^T)^{-1}(A-A^T)$:

$$\gamma_j = 1 - \frac{1}{2\alpha} - \frac{1}{2\alpha}\lambda_j \qquad (2.13)$$

For $\lambda_{max} = \max_j |\lambda_j|$ it follows that the choice

$$\alpha = \alpha_{opt} = \frac{1}{2} + \frac{1}{2}\lambda^2_{max} \qquad (2.14)$$

leads to the expression

$$\gamma_{max} = \frac{\lambda^2_{max}}{1+\lambda^2_{max}} \qquad (2.15)$$

where $\gamma_{max} = \max_j |\gamma_j|$.

From (2.15) it follows that we may expect convergence always, if $\alpha = \alpha_{opt}$, since $\gamma_{max} < 1$. It follows immediately from (2.14) that if $A=A^T$, we have $\alpha_{opt} = \frac{1}{2}$, since $\lambda_{max} = 0$. This agrees with our earlier observation that the splitting I (2.2), which results from $\alpha = \frac{1}{2}$, is optimal for almost symmetric matrices. The splitting II (2.3) results from (2.10) if we choose $\alpha = 1$, and this choice is optimal if $\lambda_{max} = 1$.

The next observation is that for systems that arise from problems like (2.1), we have in general $\lambda_{max} = O(h)$, where h is a measure for the gridsize over which is discretised. From the definition of α_{opt} , it follows that $\gamma_{max} = O(h^2)$ This indicates that the proper choice of α_{opt} yields considerably faster convergence of the corresponding iterative method (2.11). The last observation is that the eigenvalues γ_j are all situated on the straight line $X = 1 - \frac{1}{2\alpha}$ in symmetric positions to the X-axis. One could use this fact in order to follow Manteuffel's ideas for acceleration of the iterative method (2.11).

3. Outer and inner iterations.

In each step of the iterative method (2.11) a linear system of the form

$$(A+A^T)y = b' \tag{3.1}$$

has to be solved.

Such a step will be called an outer iteration step. If the linear system (3.1), arising in each outer iteration step, is solved by an iterative method, then the steps of this method will be ~~mentioned~~ described as an inner iteration steps.

In this section we will consider iterative methods, for solving (3.1), that are based on regular splittings /5/

$$A+A^T = K-R \tag{3.2}$$

For these regular splittings, it holds $K^{-1} \geq 0$ and $R \geq 0$ and moreover, if $R \neq 0$,

$$0 < K^{-1} < (A+A^T)^{-1} \tag{3.3}$$

One might follow different strategies in the outer-inner-iteration process. the most extremal strategies are considered here in some detail.

Strategy 1. The equation (3.1) is solved at each outer-iterationstep accurately, that is, with an accuracy less than or equal to the desired accuracy in the final solution of Ax=b.

Strategy 2. Only one step of the inneriteration process to solve (3.1) is performed at each outer-iteration step.

It might be expected that strategy 2 leads to an increase in the number of outer-iterations, needed to reach a certain accuracy, as compared to strategy 1. From a practical point of view however, it is interesting to investigate whether the total number of inneriterations decreases.

In the numerical experiments (section 4) it was observed that strategy 2 was the cheapest one. It was even observed that strategy 2 needed sometimes <u>less</u> outeriterations, which is somehow in contrast with the conservationlaw of trouble. This effect will be more or less explained here for the case $\alpha = 1$. For a more general choice of $\alpha > 0$, the effects can be explained similarly.
For strategy 2 it follows from (3.2) that

$$K x_{i+1} = b + A^T x_i + R x_i \qquad (3.4)$$

and, if we set $R = K - (A + A^T)$ and $h_i = x_i - x$, then we have

$$K h_{i+1} = \left[A^T + K - (A + A^T) \right] h_i \qquad (3.5)$$

or, more conveniently

$$h_{i+1} = \left[I - \tfrac{1}{2} K^{-1} (A + A^T) - \tfrac{1}{2} K^{-1} (A - A^T) \right] h_i \qquad (3.6)$$

We now assume, in a very rough and not precise way that

$$K^{-1} = \kappa (A + A^T)^{-1} \qquad (3.7)$$

and from (3.3) we have $\kappa < 1$.
Equation (3.6) can now be rewritten in

$$h_{i+1} = \left[(1 - \tfrac{1}{2}\kappa) I - \tfrac{1}{2}\kappa (A + A^T)^{-1} (A - A^T) \right] h_i \qquad (3.8)$$

For the eigenvalues δ_j of the matrix $(1 - \frac{1}{2}\kappa) I - \frac{1}{2}\kappa (A + A^T)^{-1} (A - A^T)$ we have that

$$\delta_j = 1 - \tfrac{1}{2}\kappa + \tfrac{1}{2}\kappa \lambda_j \qquad (3.9)$$

For strategy 1, we have $\kappa = 1$, and thus $\delta_j = \gamma_i$. It may be expected that strategy 2 needs less outeriterations if $\delta_{max} < \gamma_{max}$, where $\delta_{max} = \max_j |\delta_j|$.
Let therefore $\tilde{\lambda}$ be defined as [redacted] the solution of

$$\delta_{max} = \gamma_{max} \quad (\alpha = 1 , \kappa < 1) \qquad (3.10)$$

From simple calculations, it follows that

$$\tilde{\lambda} = \frac{(2 - \kappa)^2 - 1}{1 - \kappa^2} \qquad (3.11)$$

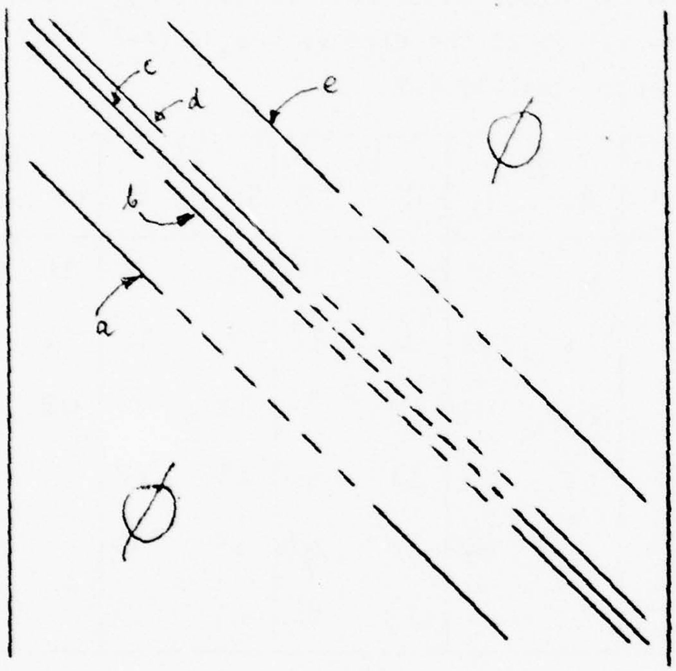
As in general κ will be very near to 1, we have $\tilde{\lambda} \approx 1$. This explains somehow that in strategy 2 fewer [redacted] outeriterations are necessary if $\lambda_{max} > 1$. However, it should be stressed [redacted], that for $\lambda_{max} < 1$, strategy 1 needs fewer [redacted] outeriterations indeed, but often far more inneriterations.

4. Numerical experiments.

The matrices in the experiments can be considered as to [redacted] arising from the five-point discretisation of the partial differential equation

$$\Delta u + \beta u_x = 0 \qquad , \qquad (4.1)$$

where u_x is discretised by a central difference formula. The equation (4.1) is given over a rectangular region, and u is equal to a given function along the boundaries. The matrix arising thus, has the structure



(4.2)

The elements of the diagonals are denoted by a_i, b_i, c_i, d_i and e_i, where i is counted rowwise. As a regular splitting for $A+A^T$ we choose the incomplete decomposition that arises in the ICCG(3)-method /4/. In both the examples, (4.1) has been discretised over a rectangular grid with 30 meshpoints in the x-direction as well as in the y-direction. This yielded a matrix A of order 900 and with half bandwith 30.

Example 1. In the first example we demonstrate some of the effects already described. Therefore β , the constant in differential equation (4.1) has been chosen such, that the non-zero values of the elements in (4.2) were:

$a_i = -1.$, $b_i = -1.25$, $c_i = 4.$, $d_i = -.75$, $e_i = -1.$

In table I the results for different strategies are listed. The number of outeriterations and the total number of inner-iterations to reach a certain precision are represented. The precision was estimated by $\max_i |x_{ji} - x_{j+1\,i}|$, where x_{ji} is the i-th element of the outeriterationvector x_j . From rather rough calculations it followed that λ_{max} (the max. of the absolute values of the eigenvalues of $(A+A^T)^{-1}(A-A^T)$ has a value of approximately 1.7.

precision	I		II		III		IV		V		VI	
	A	B	A	B	A	B	A	B	A	B	A	B
10^{-1}	after 140 outeriterations (≈1400 inner-) only little improvement. Convergence?		4	10	4	10	12	12	31	31	not a convergent process!	
10^{-2}			19	32	16	28	18	18	39	39		
10^{-3}			30	47	24	36	23	23	48	48		
10^{-4}			38	57	32	46	28	28				
10^{-5}			46	67	40	56	33	33				
10^{-6}					47	65	37	37				

Table I. Results for different strategies

Explanation ~~to~~ of table I:

The numbers I up to VI stand for the different strategies. The number of outeriterations is counted by A and B represents the total number of inneriterationsteps. The strategies were:

I: $(\alpha = 1)$. For each outeriterationstep, the inneriterationprocess was performed in an accurate way (with a maximum of 10 steps). This is <u>strategy 1</u> (conv. for $\lambda_{max} < \sqrt{3}$).

II: $(\alpha = 1)$. The inneriterationprocess was stopped as soon as the residual was less than 10% of the initial residual at each outeriterationstep (in $\| \ \|_2$ -norm).

III: $(\alpha = 1)$ The inneriterationprocess was stopped as soon as the residual was less than 20% of the initial residual.

IV: $(\alpha = 1)$ <u>Strategy 2.</u>

V: $(\alpha = \frac{1}{2})$ <u>Strategy 2.</u>

VI: $(\alpha = \frac{1}{2})$ <u>Strategy 1.</u> (convergence for $\lambda_{max} < 1$)

A few additional remarks:

1. As $\lambda_{max} \lesssim \sqrt{3}$ it could be expected that strategy I resulted in a convergent process and strategy VI in a divergent process.
2. It is surprising that strategy V ($\alpha = \frac{1}{2}$) leads to a convergent process.
3. The ideas in section 3 are underlined by the results of strategies I, II, III and IV.

<u>Example 2.</u> In order to demonstrate some effects for a smaller λ_{max}, β was chosen such that the following values for the nonzero elements in (4.2) resulted:

$a_i = -1.$, $b_i = -1.1$, $c_i = 4.$, $d_i = -0.9$, $e_i = -1.$

From rough computation, it was estimated that $\lambda_{max} \simeq .7$ (< 1). In table II the results are given for <u>strategy 1</u> (accurate inneriteration) and <u>strategy 2</u> (1-step inneriteration), both for the choice $\alpha = 1$ in (2.11).

precision	strategy 1		strategy 2	
	A	B	A	B
10^{-1}	2	20	8	8
10^{-2}	6	55	13	13
10^{-3}	10	83	18	18
10^{-4}	14	101	23	23
10^{-5}	19	115	29	29
10^{-6}	25	121	34	34

Table II. Results for example 2.
A = number of outeriterations.
B = total number of inneriterations.

As could be expected, we see that the number of outeriterations increases from strategy 1 to strategy 2. From the other side we see that the total number of inneriterations decreases sharply, and thus the total computing time too.

5. Final remarks.

It was observed in practical situations that the choices $\alpha = \frac{1}{2}$ or $\alpha = 1$ in the iterative method (2.11), in combination with a 1-step strategy for the inneriteration often lead to a fairly efficient and easy to program method. Only the inneriteration based on an incomplete choleski-factorization (see ICCG(3) in /4/) has been considered. No attempts have been made to accelerate (2.11), nor has it been tried to estimate λ_{max} in order to choose an optimum value α_{opt} . It should be mentioned that for strategy 1, the eigenvalues γ_j of the iterationmatrix are also situated on a line parallel to the X-axis in the complex plane, while in strategy 2 the eigenvalues δ_j are not situated on such a line, but in the neighbourhood of it.

ACKNOWLEDGEMENTS

This work was supported in part by U.S. Army Research & Development, under Grant DA-ERO-75-G-084

6. References.

/1/ P. Concus & G.H. Golub, "A generalized conjugate gradient method for nonsymmetric systems of linear equations"(to app.)

/2/ O. Widlund, Tech. Rept., Courant Inst., New York (to appear)

/3/ T.A. Manteuffel, "An iterative method for solving nonsymmetric linear systems with dynamic estimation of parameters", Ph.D. Thesis, University of Illinois, 1975.

/4/ J.A. Meijerink & H.A. van der Vorst, "An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix, Math. of Comp. (to appear)

/5/ R.S. Varga, "Matrix Iterative Analysis", Prentice-Hall, 1962.

APPENDIX C

Automatic monitoring of Lanczos-schemes for symmetric or skew-symmetric generalized eigenvalue problems.

by

J.M. van Kats [+] *and* H.A. van der Vorst.

[+] Academic Computer Center, Utrecht, The Netherlands.

Contents

Introduction

Useful variants of the Lanczos scheme for the determination of eigenvalues of large symmetric matrices have been developed in the past few years (Paige [6], Golub [7], Lewis [11]). Symmetry of the matrices is essential in the Lanczos method. Some other eigenvalue problems can be reduced to symmetric problems after some preliminary work e.g. if the matrix A is skew-symmetric ($A=-A^T$) the scheme can be applied to the matrix A^2, which involves twice as much computational work (Cline [12], Lewis [11], Platzman [13]).
Another important class of problems is concerned with the determination of eigenvalues of the productmatrix CB where C and B are symmetric matrices and one of them, say B, is positive definite as well. A common way to solve this problem with the Lanczos scheme is to construct first a Choleski decomposition $B=LL^T$ and then to apply the scheme to L^TCL which has eigenvalues identical to those of CB (Golub [7]). Since Lanczos-schemes are specially attractive for sparse matrices, a disadvantage of this approach might be a loss of sparsity in the Choleski decomposition.

In section 1 of this report a generalized Lanczos scheme is proposed that applies directly to matrices A whether they are symmetric or skew-symmetric, and to productmatrices CB where C is either symmetric or skew-symmetric and B is symmetric positive definite.
The matrices A, B and C do not have to be represented in the usual way as two-dimensional arrays of numbers, but as rules to compute the products Ax, Bx, and Cx for any given x. This allows us to take full advantage of any sparsity structure.

Lanczos schemes yield approximations for the eigenvalues of the given eigenvalueproblem. One of the main difficulties is how to distinguish between good and bad approximations, since both are generally present (Paige [6], Parlett and Kahan [2]). In section 2 an algorithm is proposed to determine the good approximations and to remove the bad ones. It should be mentioned here that any multiplicity of an eigenvalue of the matrix can not be detected. A multiplet, if there is one, will be represented by only one single eigenvalue; this problem is peculiar to Lanczos schemes (Kahan and Parlett [2]).

In section 3 an implementation of the algorithm of section 2 is given. Numerical examples for the algorithms of both sections 1 and 3 are presented in section 4.
We did not consider in detail the problem of determining of eigenvectors. In section 5 we summarize the main results of Kahan and Parlett [2] as well as some of our numerical results.

Fortran subroutines for both the generalized Lanczos scheme and the detection of good eigenvalue approximations are covered in the final section of this report.

1. A generalized Lanczos scheme

In this section we describe Lanczos schemes that apply to skew-symmetric matrices or product-matrices CB, where B is symmetric positive definite and C is either symmetric or skew-symmetric (all the matrices A, B and C will be of order n). It should be mentioned here that the eigenvalues of CB are identical to those of BC.

Eigenvalue problems $Cx=\lambda Bx$ can be reduced to either of these forms.

The algorithms are closely related to an algorithm published by Widlund [5] for the solution of non-symmetric linear systems.

The eigenvalue problems of a skew-symmetric matrix A can be reduced to the eigenvalue problem of a symmetric matrix by squaring the matrix: A^2. This is not necessary in our formulation, Definition of the generalized Lanczos scheme

Let A be of the form A=CB, where B is symmetric positive definite and C is either symmetric or skew-symmetric.
Choose an arbitrary vector v_1, with $(v_1,v_1)_B=1$, and form $u_1=Av_1$.
Rows $\{v_j\}$, $\{\alpha_j\}$, $\{\beta_j\}$ and $\{\gamma_j\}$ are then generated by

$$\left.\begin{aligned} \alpha_j &= (v_j\ ,\ Av_j)_B \\ w_j &= u_j - \alpha_j v_j \\ \gamma_{j+1} &= (w_j\ ,\ w_j)_B^{\frac{1}{2}} \\ \beta_{j+1} &= \tau\gamma_{j+1} \\ v_{j+1} &= \frac{1}{\gamma_{j+1}} w_j \\ u_{j+1} &= Av_{j+1} - \beta_{j+1} v_j \end{aligned}\right\} \begin{array}{l} j=1,2,\ldots,m \\ (\text{as fas as } \gamma_j \neq 0, \\ \ \text{see note 1)} \end{array} \tag{1.1}$$

where $(x,y)_B=(x,By)$, with B symmetric and positive definite,
and $\tau=1$ if $C=C^T$
$\tau=-1$ if $C=-C^T$

(see also note 2)

The constants α_i, β_i and γ_i define a tridiagonal matrix T_m:

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & \emptyset \\ \gamma_2 & \alpha_2 & \beta_3 & & \\ & & & & \\ & \emptyset & & & \beta_m \\ & & & \gamma_m & \alpha_m \end{pmatrix}$$

Note 1: If in some stage $\gamma_j=0$, then one can either restart with a new v, or proceed with γ_j replaced by some small constant. In practice the situation $\gamma_j=o$ occurs very incidentally. In our implementation such a γ_j is replaced by a small constant.

Note 2: For B=I and $\tau=1$ we have the original Lanczos scheme as defined by Paige.

Theorem

We assume that either $C=C^T$ $(\tau=1)$ or $C=-C^T$ $(\tau=-1)$ holds and that B is a positive definite symmetric matrix and A=CB, then the generalized Lanczos scheme applied to A generates a tridiagonal matrix T_m, where limit-values of the eigenvalue of T_m for increasing m, should be equal to some of the eigenvalues of A, but they may differ by a certain amount depending on the precision of computation.

Proof

i) For $C=C^T$ and B=I, the result is well known (Paige [6], Golub [7]).

ii) For $C=-C^T$and B=I the proof is as follows:
It is only necessary to establish that the generated row $\{v_k\}_{k=1..m}$ is an orthonormal row. The proof is by induction. Let $\{v_k\}_{k=1,...,j}$ be an orthonormal row.
Then we have for v_{j+1} the relation:

$$\gamma_{j+1}v_{j+1} = Cv_j - \beta_j v_{j-1} - \alpha_j v_j \quad ,$$

where we assume that $\gamma_{j+1}\neq 0$, since in that case the recurrence relation terminates.

For k<j-1:

$$(\gamma_{j+1}v_{j+1},v_k) = (Cv_j - \beta_j v_{j-1} - \alpha_j v_j,v_k)$$

$$= -(v_j,Cv_k)$$

$$= -(v_j,\gamma_{k+1}v_{k+1} + \beta_k v_{k-1} + \alpha_k v_k)$$

$$= 0$$

For k=j-1:

$$(\gamma_{j+1}v_{j+1},v_{j-1}) = (Cv_j,v_{j-1}) - \beta_j(v_{j-1},v_{j-1})$$

$$= (Cv_j,v_{j-1}) - \beta_j$$

Since $\beta_j = -\gamma_j = -(\gamma_j v_j,v_j) = -(Cv_{j-1},v_j) = (Cv_j,v_{j-1})$

it follows that $(\gamma_{j+1}v_{j+1},v_{j-1}) = 0$

For k=j:

$$(\gamma_{j+1}v_{j+1},v_j) = (Cv_j,v_j) - \alpha_j = 0$$

Finally, we have

$$(v_{j+1},v_{j+1}) = \frac{1}{\gamma_{j+1}^2}(Av_j - \beta_j v_{j-1} - \alpha_j v_j, Av_j - \beta_j v_{j-1} - \alpha_j v_j)$$

$$= \frac{1}{\gamma_{j+1}^2}(u_j - \alpha_j v_j, u_j - \alpha_j v_j)$$

$$= \frac{1}{\gamma_{j+1}^2}(w_j,w_j)$$

$$= 1$$

Thus the row $\{v_k\}_{k=1,\dots,j+1}$ is an orthonormal row.

iii) When $C=C^T$ and B is symmetric positive definite, B can be written as $B=LL^T$, where L is lowertriangular. Since the eigenvalues of CB are equal to those of L^TCL, the original Lanczos scheme might be applied to L^TCL (with normal inner-product (,)).

In this case we then have the special relation

$$\alpha_j = (v_j, L^TCLv_j)$$

and

$$u_{j+1} = (L^TCLv_{j+1} - \beta_{j+1}v_j)$$

it follows that

$$Lu_{j+1} = LL^TCLv_{j+1} - \beta_{j+1}Lv_j$$

If we replace x by $L^T\tilde{x}$, this equation can be rewritten:

$$LL^T\tilde{u}_{j+1} = LL^TCLL^T\tilde{v}_{j+1} - \beta_{j+1}LL^T\tilde{v}_j$$

$$\tilde{u}_{j+1} = CB\tilde{v}_{j+1} - \beta_{j+1}\tilde{v}_j$$

$$= A\tilde{v}_{j+1} - \beta_{j+1}\tilde{v}_j$$

The other Lanczos relations follow from

$$\alpha_j = (L^TCLv_j, v_j)$$

$$= (L^TCLL^T\tilde{v}_j, L^T\tilde{v}_j) = (CB\tilde{v}_j, B\tilde{v}_j)$$

$$= (A\tilde{v}_j, \tilde{v}_j)_B$$

$$\beta_{j+1}^2 = \gamma_{j+1}^2 = (w_j,w_j) = (L^T\tilde{w}_j,L^T\tilde{w}_j)$$

$$= (B\tilde{w}_j,\tilde{w}_j) = (\tilde{w}_j,\tilde{w}_j)_B$$

The relations

$$\tilde{w}_j = \tilde{u}_j - \alpha_j\tilde{v}_j$$

and

$$\tilde{v}_{j+1} = \frac{1}{\gamma_{j+1}}\tilde{w}_j$$

are obvious.

The vectors $\tilde{w}_j$, $\tilde{v}_j$ and $\tilde{u}_j$ produce the desired result.

iv) The remaining case $A=CB$, where $C=-C^T$ and B is symmetric positive definite, follows from the previous ones (with $\tau=-1$). //.

Remarks

1. If $C=-C^T$, we have $\alpha_j=0$ for all j.
2. The above theorem allows the computation of the eigenvalues of CB, which are equal to those of BC, without the explicit need for an LL^T-factorization of the matrix B. This makes the new schemes very attractive, especially if B has a sparse structure. However, it should be noted that eigenvectors cannot be computed by these schemes directly, since then an LL^T-factorization is required for a proper transformation.
   Eigenvectors may be computed by a Raleigh-quotient iteration scheme [1], once one has a (fast) solver for systems like $Bx=y$. For sparse matrices B, for which fast direct or iterative solution schemes exist, this Lanczos scheme can also be used for determining eigenvalues of $Cx=\lambda Bx$, via $B^{-1}Cx=\lambda x$. The scheme should be applied to CB^{-1} which has identical eigenvalues.

3. We should like to mention briefly certain aspects of programming. For the generalized problem the adapted schemes (1.1) require only one extra matrix-vector multiplication and only one additional vector to store $B\tilde{w}_j$. Remember that $B\tilde{v}_j$ can be computed from

$$B\tilde{v}_j = \frac{1}{\gamma_{j+1}} B\tilde{w}_j$$

4. If C is skew-symmetric ($\tau=-1$) then the generated matrices T_m are also skew-symmetric. Eigenvalues of a tridiagonal skew-symmetric matrix can be computed as follows:
the matrix iT_m is Hermitian and has real eigenvalues. Since in the computation of the eigenvalues with Sturm-sequences, only squares of off-diagonal elements, β_j^2, are involved, these eigenvalues can be computed without any complex computation. Once the eigenvalues of $|T_m|$:

$$(1.3) \qquad |T_m| \equiv \begin{pmatrix} 0 & \beta_2 & & & \\ \beta_2 & 0 & \beta_3 & & \\ & & & & \\ & & & & \beta_m \\ & & & \beta_m & 0 \end{pmatrix}$$

have been computed, they should be multiplied by i so that they represent the eigenvalues of T_m.

2. Monitoring of the Lanczos process

In the single vector Lanczos processes, as described in section 1, rows of mutually orthonormal vectors are generated.
The coefficients α_j, β_j constitute a tridiagonal matrix T_m the eigenvalues of which bear some relation to those of A.
We only consider here the symmetric case ($\tau=1$), the other one ($\tau=-1$) is obvious.

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \emptyset & \\ & \beta_3 & \alpha_3 & \beta_4 & & & \\ & & & & \cdot & & \\ & & & & & \cdot & \\ & & & & & \beta_{m-1} & \\ & \emptyset & & & \beta_{m-1} & \alpha_{m-1} & \beta_m \\ & & & & & \beta_m & \alpha_m \end{pmatrix} \tag{2.1}$$

For B=I, T_m is related to A by

$$AV_m = V_mT_m + \beta_{m+1}v_{m+1}\,e_m^T \tag{2.2}$$

where V_m is an orthonormal n*m matrix, in which the v_i are the columns, and $e_m^T=(0,0,\ldots 0,1)$, the m-th unitvector (n is the order of the matrix A).
The relation of the eigenvalue-problem of T_m to the eigenvalue-problem of A is discussed and demonstrated by extensive numerical experiments reported by van Kats and van der Vorst [3].

2a. The eigenvalue problem $T_m x=\lambda x$.

The eigenvalues of T_m are the roots of

$$\det(T_m - \lambda) = 0 \tag{2.3}$$

If the leading k-th order principal minor of T_m is denoted by T_k, then the following recurrence relation holds

$$\det(T_k - \lambda) = (\alpha_k - \lambda)\det(T_{k-1} - \lambda) - \beta_k^2 \det(T_{k-2} - \lambda) \tag{2.4}$$

If we define $\det(T_0-\lambda)=1$ and since we have $\det(T_1-\lambda)=\alpha_1-\lambda$, it follows that the above relationship is exactly that of orthogonal polynomials

$$p_k(x) = (\alpha_k - x)\, p_{k-1}(x) - \beta_k^2 p_{k-2}(x) \tag{2.5}$$

It follows that the zeros of p_k separate the zeros of $p_{k-1}(x)$ and $p_{k+1}(x)$ in a strict sense if none of the β_k equals zero (Wilkinson [1]).

By analogy then, if the ordened eigenvalues of T_k are denoted by $\lambda_i^{(k)}$, we have

$$\lambda_{i-1}^{(k-1)} < \lambda_i^{(k)} < \lambda_i^{(k-1)} \tag{2.6}$$

2b. Recognition of limitvalues

From relation (2.6) it follows, that the extreme eigenvalues of T_m, for increasing m, converge strictly monotonically. Since according to Paige limitvalues of the row T_m are equal to eigenvalues of the original matrix A, except for some amounts that depend on the precision of computation, this property may be used for an automatic determination of the extreme eigenvalues. However, it is evident that limit sequences can also be recognized for internal

eigenvalues, since the strict separation relationship (2.6) should hold.
As soon as relation (2.6) is violated, either because $\lambda_i^{(k)}$ equals one of the $\lambda_{i-1}^{(k-1)}$ or $\lambda_i^{(k-1)}$, or is outside the interval $[\lambda_{i-1}^{(k-1)}, \lambda_i^{(k-1)}]$ we know that at least in the precision in which we are working it is not possible to distinguish between $\lambda_i^{(k)}$ and the respective eigenvalue of T_{k-1}. Consequently we have a limitvalue and thus an approximate eigenvalue of A. Since we are working in a finite precision and the extreme eigenvalues are bounded by the extreme eigenvalues of A, the separation relationship (2.6) will be violated sooner or later.
In practice this provides us with an excellent means of recognizing limitvalues automatically.

As soon as one case of violation has been encountered one measures how much the respective values, say b and c, differ relatively.
A value ε_{bc} is defined as follows

$$\varepsilon_{bc} = \frac{\text{abs}(b-c)}{1+\text{abs}(b)} \tag{2.7}$$

The maximum over all violations will be taken as ε. This ε yields an impression of the relative accuracy with which all the eigenvalues T_k and T_{k-1} have been computed.
(N.B.: this is not to be confused with the accuracy of the Lanczos-process itself).
As a rule of thumb this ε is multiplied by n (the order of the original matrix A) and all eigenvalues of T_k and T_{k-1} which differ by less than $n*\varepsilon$ will be taken as possible limitvalues.
In the next section this will be stated more precisely.

3. EVSCAN; an implementation of the monitoring process

The monitoring process is essentially based on the separation relationship, as described in the previous section. This requires the computation of all the eigenvalues of two succeeding tridiagonal matrices T_{k-1} and T_k.
At the first stage of the process one checks to see whether the separation relationship has been violated. It is well known that in the Lanczos-process multiplets of eigenvalues are introduced as soon as orthogonality has been lost [3].
The next step is to recognize these multiplets. Each multiplet will be represented by one single eigenvalue interval. If the original matrix A has a multiplet eventually, this will not be recognized.
After the eigenvalues of T_{k-1} and T_k have been scanned for multiplets, the resulting multiplet-free rows are compared in order to determine those limitvalues, which represent eigenvalues of A.
The monitoring process will be described in detail below.

Step 1:

Check whether the eigenvalues $\lambda_i^{(k)}$ of T_k separate the eigenvalues $\lambda_i^{(k-1)}$ of T_{k-1} in a strict sense. If some $\lambda_i^{(k)}$ is outside the interval $[\lambda_{i-1}^{(k-1)},\lambda_i^{(k-1)}]$ then this yields a lowerbound ε' for the highest attainable relative precision in all the eigenvalues, and we define $\varepsilon''=\max \varepsilon'$, where the maximum is taken over all violations. If no violation has been encountered then ε'' is taken to be 2^{-t}, where t is the number of digits in floating point arithmetic.
An upperbound for the relative working precision, to be used in the following steps, is estimated by

$$\varepsilon = n * \varepsilon''$$

where n is the order of the matrix A.

Step 2:

With the ε resulting from step 1, the row $\{\lambda_i^{(k-1)}\}$ and $\{\lambda_i^{(k)}\}$ are both scanned separately for multiplets. As soon as a multiple value has been discovered, i.e. two values are encountered which differ relatively by less than ε, the eigenvalue concerned is represented as an interval with the smallest value of the multiplet as the lowerbound of the interval and the largest one of the upperbound. If successive eigenvalues are recognized as belonging to the same multiplet, this may lead to a larger value for the relative precision $\varepsilon \quad \left(\frac{\text{abs(upperbound-lowerbound)}}{\text{1+abs(lowerbound)}} \right)$.

Step 2 is repeated with the most recent value of ε as long as ε increases.

Step 3:

From step 2 two rows of intervals result, representing eigenvalues of T_{k-1} and T_k respectively. These rows are, as far as possible, multiplet-free with respect to ε.

For each interval in one row one checks to see whether there is an interval in the other row that intersects with the first one or is at a distance of less than ε relatively. If one of these conditions has been met, a new interval is chosen as the span of both. The length of the new interval yields a new value for ε.

Step 3 is repeated with the most recent value of ε as long as ε increases.

If 6 successive intervals in this process are encountered belonging to T_{k-1} or T_k for which the above condition does not hold, then a hole in the spectrum is assumed. The value 6 has been chosen from numerical experience and could be replaced by any better value.

Continuing in this fashion, step 3 delivers one single row of intervals, each of which may be considered to contain a limitvalue. These limitvalues differ only from the eigenvalues of the original matrix A with regard to the degree of precision in which we are computing.

C U -DIMENSION U(N). SCRATCH-ARRAY.

In some situations it may occur that step 3 yields an interval which contains no eigenvalue of A. However in such cases there is an eigenvalue in the neighbourhood of the interval. In these situations it is common for the process to yield also the interval in which the respective eigenvalue is situated; thus two very close intervals are obtained.

In order to identify both intervals as representing the same eigenvalue, it is advisably to apply only step 2 to the final row with a slightly larger value for ε (10*ε,say).

```
C                 PROGRAM LNCZOS(OUTPUT)
C                 DIMENSION QQ(21),Q1(21),U(21),ALPHA(16),BETA(16)
```

4. Numerical Experiments

All the numerical experiments have been carried out on the CDC Cyber 73-28 of ACCU. The relative working precision is 48 bits (about 14 decimal digits).

4.1 The Bar-problem

The bar-problem is known to cause difficulties in the determination of the eigenvalues at the lower end of the spectrum [4, 3]
The 40th order matrix is given by:

$$\begin{pmatrix}
5 & -4 & 1 & & & & & & \\
-4 & 6 & -4 & 1 & & & \emptyset & & \\
1 & -4 & 6 & -4 & 1 & & & & \\
 & & & & & & & & \\
 & & & & & & & & \\
 & \emptyset & & & 1 & -4 & 6 & -4 & 1 \\
 & & & & & 1 & -4 & 6 & -4 \\
 & & & & & & 1 & -4 & 5
\end{pmatrix}$$

With EVSCAN the eigenvalues are determined in each 10th step of the Lanczos-process i.e. T_{10*k} is compared to T_{10*k-1}. In table I we summarize the number of different eigenvalue intervals, as delivered by EVSCAN for k=3 up to 40. We note that when k is 29, 30 or 37 more than 40 eigenvalues are found. If we follow the advice and make an extra scan, as described in section 2, then in general the number of eigenvalue intervals is not affected, except when k is 29, 30 or 37. In the latter cases the final number of eigenvalues was corrected to 40 at the cost of some slightly larger intervals. This implies that there have been migrating eigenvalues very close to an eigenvalue.

From the experiments it follows that all eigenvalues, including those at the lower end of the spectrum, are detected by EVSCAN at some stage. In order to determine the eigenvalues at the lower end one has to perform a number of Lanczos steps larger than the order of the matrix (70 steps; order is 40).

From table I it can be seen that the estimated ε, scaled with respect to the machine-accuracy, increases slowly. Since it is natural to relate the accuracy to the order of the respective tridiagonal matrices it can be seen from the fourth column of table I (ε/k) that the eigenvalue determination with this process is rather stable.

4.2 Wilkinson's W_{21}^{+} and W_{21}^{-}

Wilkinson [1] has introduced two classes of tridiagonal matrices W_{2n+1}^{+} and W_{2n+1}^{-}.

W_{2n+1}^{+} is defined by the relations

$$\alpha_i = n+1-i \quad (i=1,\ldots n+1)\ , \qquad \alpha_i = i-n-1 \quad (i=n+2,\ldots 2n+1)$$
$$\beta_i = 1 \quad (i=2,\ldots 2n+1)$$

and W_{2n+1}^{-} by the relations

$$\alpha_i = n+1-i \quad (i=1,\ldots n+1)\ , \qquad \alpha_i = n+1-i \quad (i=n+2,\ldots 2n+1)$$
$$\beta_i = 1 \quad (i=2,\ldots 2n+1)$$

The matrices W_{21}^{+} and W_{21}^{-} have been used for our tests.

$$W_{21}^{+} = \begin{pmatrix}
10 & 1 & & & & & & & \\
1 & 9 & 1 & & & & \emptyset & & \\
 & 1 & 8 & 1 & & & & & \\
 & & & & & & & & \\
 & & & 1 & 0 & 1 & & & \\
 & & & & & & & & \\
 & \emptyset & & & & 1 & 8 & 1 & \\
 & & & & & & 1 & 9 & 1 \\
 & & & & & & & 1 & 10
\end{pmatrix}$$

$$W_{21}^{-} = \begin{pmatrix} 10 & 1 & & & & & & & \\ 1 & 9 & 1 & & & & & \emptyset & \\ & 1 & 8 & 1 & & & & & \\ & & & 1 & 0 & 1 & & & \\ & & & & & 1 & -8 & 1 & \\ & \emptyset & & & & & 1 & -9 & 1 \\ & & & & & & & 1 & -10 \end{pmatrix}$$

Table II gives the eigenvalues of W_{21}^{+} and table III those of W_{21}^{-}.

As can be seen from table II some of the eigenvalues of W_{21}^{+} are quite close. With the precision in which we are computing we cannot expect to detect 21 separate eigenvalues; at least λ_{21} and λ_{20} might behave as multiple eigenvalues.
Since the error in the Lanczos-process increases slowly (see 4.1), W_{21}^{+} has been chosen in order to determine which eigenvalues are recognized as being distinct in several stages of the process. Also we may get some impression of the behaviour of the Lanczos-process for (almost) multiple eigenvalues. The motivation for the choice of W_{21}^{-} is somewhat different. The eigenvalues of W_{21}^{-} are equal and opposite in pairs. It is well known that the power method gives slow convergence in such cases [1]. Since there is some relationship between the powermethod and the Lanczos-method it might be interesting to apply the Lanczos-process to this matrix.

With respect to the numerical experiments for W_{21}^{+}, the following observations have been made.

- From a selection of the results, as represented in table IVa, it follows that in no case all 21 eigenvalues are detected. Only for m=32, 20 eigenvalue intervals have been determined, for higher values of m even λ_{19} and λ_{18} are represented by one eigenvalue interval. This explains also the increase in ε.

- A more serious defect is the following. With almost multiple eigenvalues, which are distinct within our accuracy of computation (λ_{19}, λ_{18} and λ_{17}, λ_{16}) eigenvalue intervals are delivered which are in between both eigenvalues. These eigenvalue intervals differ significantly from both true eigenvalues, see table Va (eigenvalue .8038.....E+01 and .9210.....E+01) and compare these values with those in table II.
- Table Va-d list detailed results for W^{+}_{21}.

With respect to W^{-}_{21} it is observed that there are no problems in determining the eigenvalues, except with respect to the speed of convergence. For m=16 (see table IVb) no eigenvalues have been detected automatically.
Detailed results are listed in table VIa-c.

4.3 Pathologically clustered eigenvalues

It is well-known that the convergence properties depend highly on the relative clustering of the eigenvalues. Therefore we have constructed a matrix with a cluster of eigenvalues at each end of the spectrum and one single eigenvalue in between both clusters.

The matrix A chosen was a 40th order diagonal matrix with diagonal
elements: 1.001, 1.002, ..., 1.019, 2.000,
3.021, 3.022, ..., 3.039, 3.040.

After 10, 20, 30 steps of the Lanczos-process only the eigenvalue 2.0 is determined automatically (no convergence at the lower and upper end of the spectrum).
After 40 Lanczos steps convergence was signalled at both lower and upper end of the spectrum but the second eigenvalue of A (1.002) was not found. For results see table IIIa.
After 45 Lanczos steps all eigenvalues have been determined, see table VIIb.

4.4 A large full matrix

The Lanczos-method is proposed usually for the determination of the extreme eigenvalues of *sparse* matrices. We have used the Paige style Lanczos algorithm to compute some of the extreme eigenvalues of a symmetric full matrix of high order (n=519).

The matrix, used in this example, originates from a nuclear shell-model calculation [8].
In such a calculation one computes the matrix elements of a given one plus two-body interaction Hamiltonian in a set of j-j coupled many particle basis-states. After diagonalisation one obtains the energies and the wave-functions of the systems. In the present case the basis chosen is approximate for the description of nuclear states in $^{56}N_i$ with zero angular momentum and positive parity.
The order of the matrix is 519; it contains about 58% zero-valued elements. No advantage has been taken of the zero values which are distributed in rather an irregular way.
This matrix has well separated eigenvalues (no multiplets), which are distributed over the interval (-160.0, -180.5).

In general the eigenvalue problem for full symmetric matrices is solved by the Householder method [1].
For matrices which cannot be stored in fast core, this process is complicated to programme. Since the CP-time used is roughly proportional to $1/3\ n^3$ (n is the order of the matrix), the Lanczos-process for the determination of the extreme eigenvalues is advantageous if less than 1/3 n iterations are required.
Other practical advantages of the Lanczos-process in this case are that it is easy to restart and easy to programme.

In table VIIIa-b the results are listed for 40 and 60 Lanczos steps respectively; in the latter case the scanning process has been performed with a larger ε too (table VIIIc).
A larger ε has been chosen since the process for a large full matrix

is generally expensive and one wants to extract as much information as possible from the iteration-steps performed. The possibility of scanning with a larger ε has not been included in EVSCAN but could be with a minor modification.

For results see table VIIIa-c. For m=40, convergence at the lower end of the spectrum is detected, for m=60 convergence is detected at both the lower and the upper end.

4.5 A large sparse matrix

For the matrix used here, we have chosen the modified Laplace problem as describe in [14].

The matrix A results from five point discretisation of $\Delta u=0$ over the square region $0\leq x$, $y\geq 1$. The boundary conditions are $\frac{\partial u}{\partial u}=0$ for x=0 x=1 and y=1, and u=1 for y=0.

This equation was discretised over a rectangular grid with meshspacings $h_x=\frac{1}{31}$ and $h_y=\frac{1}{32}$, thus yielding a matrix of order 992. For this matrix the ICCG(3) decomposition

$$A = L_3L_3^T + R_3$$

is constructed (see [9]). The eigenvalues of $L_3^{-1}AL_3^{-T}$ have to be computed. They are very strongly clustered around the value 1.0. Also at the upperside of the spectrum the eigenvalue distribution is very dense. The largest eigenvalue is 1.17 and the smallest one is close to 0.0.

In table IX the results (number of eigenvalues and scaled ε) are listed for several stages of the Lanczos-process (m denotes the number of steps).

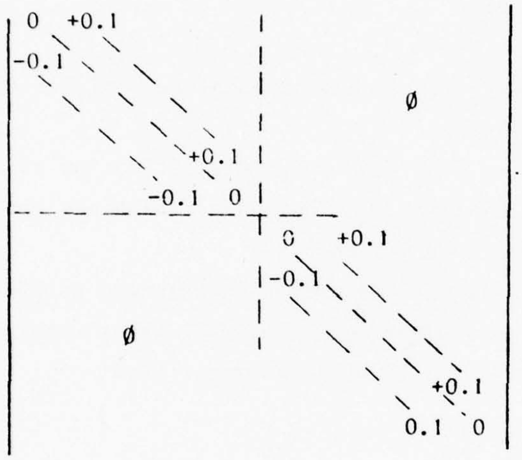
In figure I the following quantities are represented as a function of m:

- the total number of eigenvalues, detected by EVSCAN (x)
- the number of eigenvalues at the lower end of the spectrum (0). This number was fairly well represented by the parameter NDIV (see the description of EVSCAN).
- the number of eigenvalues at the upper end of the spectrum (+).

4.6 A generalized eigenvalue problem $CBx=\lambda x$

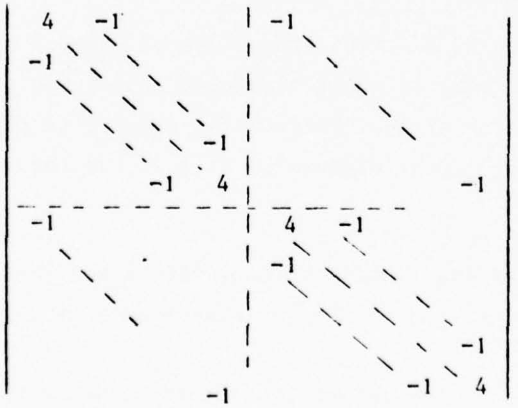
In order to demonstrate the applicability of the generalized Lanczos-scheme, as described in section 1, the following problem has been chosen. Some of the eigenvalues of CB will be computed, where B is the 5 point finite difference approximation of the Poisson-operator Δ over a square 10×10 grid and C is the central difference approximation of the operator $\frac{\partial}{\partial x}$ on the same grid. The matrices look like this

C =



blocksize is 10,
10 blocks along the
diagonal: order of
C is 100

B =



same structure
as C.

The results are listed in table X for m=15, 30, 60, where m is the order of the tridiagonal matrix, generated in the generalized Lanczos-scheme.

5. Notes on eigenvectors

5.1 Theoretical Aspects

Before we mention some results of our numerical experiments, relevant results of Kahan and Parlett [2]are summarized. The results of the Paige style Lanczos scheme for a symmetric or skew-symmetric n-th order matrix A can be written as:

$$AV_k = V_k T_k + \beta_{k+1} v_{k+1} e_k^T$$

where $e_k^T=(0,0,0,\ldots,0,1)$, the k-th unitvector, V_k is formed by the columns v_i , $i=1,\ldots,k$.

For any μ and normalized vector x, the quantity $||Ax-\mu x||$ bounds the error between μ and some eigenvalue λ of A. If μ is an eigenvalue of T_k and y the associated eigenvector then

$$\begin{aligned} |\lambda - \mu| \leq ||Ax - \mu x|| &= ||AV_k y - \mu V_k y|| \\ &= ||AV_k y - V_k T_k y|| \\ &= ||\beta_{\mu+1} v_{k+1} e_k^T y|| \\ &= |\beta_{k+1}|\ |y_k| \end{aligned}$$

Thus the error in this computed eigenvalue is bounded by $|\beta_{k+1}|$ times $|y_k|$ and if y_k, which is the last component of the vector y, is small, then the.bound may be sufficiently small even though β_{k+1} is of moderate size. From this analysis it follows that the more accurate approximations to eigenvalues may be those eigenvalues of T_k whose associated eigenvectors have rapidly dwindling components.

5.2 Numerical results

Eigenvectors of the original matrix A have been computed in the following way.
With EVSCAN eigenvalue intervals of the final tridiagonal matrix T_k have been determined. Since TSTURM (Eispack [10]) requires intervals, these intervals could be supplied directly. Since an eigenvalue of T_k could be equal to an upperbound or a lowerbound of an interval, which is not permitted by TSTURM, the intervals have been made slightly larger. The eigenvectors of T_k have to be backtransformed by V_k to eigenvectors of A.
For productmatrices an extra LL^T-decomposition of the matrix B (see section 1) is required for backtransformation, thus nullifying the advantages of the generalized Lanczos-scheme.

In the experiments we demonstrate the behaviour of the last components of a normed eigenvector of T_k. To this end, the matrix W^+_{21} (see section 4) has been chosen.

5.2.1 For k=13 no eigenvalues could be detected automatically by EVSCAN; this is reflected by the behaviour of the last components of some eigenvectors.
A Lanczos-approximation for the eigenvalue 2.130209219363 was: 2.13327....... . The last 4 components of the corresponding eigenvector of T_{13} are:

0.31 , -0.35 , -0.013 , 0.31 ,

These components also indicate that no convergence had occurred.

Also for k=13, the largest eigenvalue of W^+_{21}, 10.746194182903 has been approximated by 10.746194182902. The better convergence is reflected by the last 4 components of the corresponding eigenvector of T_{13}:

-.0041 , -.0010 , -.00013 , -.000013

evident that limit sequences can also be recognized for

5.2.2 For k=16, EVSCAN detected a number of eigenvalue intervals (see section 4).

A Lanczos-approximation $\tilde{\lambda}$ for the eigenvalue 10.746194182903 was given by the same value. The last 4 components of the corresponding eigenvector of T_{16} are:

.13E-04 , .27E-06 , .57E-09 and .59E-10.

If we denote $\tilde{x}$ as the backtransformed eigenvector of W_{21}^{+}, then we have $\dfrac{||W_{21}^{+}\tilde{x} - \tilde{\lambda}\tilde{x}\ ||_2}{||\tilde{x}||_2} = .35E-10$

In this case EVSCAN yields an eigenvalue interval which does not contain an eigenvalue of W_{21}^{+} (though there is one close by).

The eigenvalue 7.003952209528 was approximated by: 7.003952209098.

In this case the last 4 components of the eigenvector of T_{16} are:

.21E-01 , .89E-03 , .48E-05 and .15E-05.

Finally we have $\dfrac{||W_{21}^{+}\tilde{x} - \tilde{\lambda}\tilde{x}||_2}{||\tilde{x}||_2} = .86E-06.$

6. Conclusions

As far as accuracy and efficiency are concerned the generalized Lanczos-scheme applied to skew-symmetric matrices is more attractive than the original Lanczos-scheme applied to the squared matrix, with respect to both accuracy and efficiency.
For product-matrices it should be preferred because it needs no LL^T-decomposition. However special care has to be taken if eigenvectors are desired.
One of the main difficulties in using Lanczos-schemes is the monitoring of the results. This difficulty has been largely overcome by the monitoring process, described in this report. However it should be stressed that the problem of determining whether an eigenvalue of the original matrix is single or not has not been solved.

7. Programmature

Subroutines, in Fortran IV, are available of implementations of the Paige-style Lanczos and of the generalized Lanczos-schemes. These subroutines, LSVLAN and GENLAN, as well as the subroutine EVSCAN are included in the programlibrary ACCULIB of the Academic Computer Centre Utrecht.

In this section documentation and listings are given. This documentation contains a complete example of use.

```
C ********************
C HEADING 70517
C ********************
C       SUBROUTINE LSVLAN(N,IFIRST,M,RESTRT,Q1,AX,SAVEQ,U,QQ,ALPHA,BETA)
C            LOGICAL RESTRT
C            DIMENSION Q1(N),QQ(N),U(N),ALPHA(M),BETA(M)
C            EXTERNAL AX,SAVEQ
C ********************
C PURPOSE
C ********************
C       TO TRANSFORM A SYMMETRIC MATRIX A TO TRIDIAGONAL FORM T, BY ORTHOGONAL
C       TRANSFORMATIONS BY THE LANCZOS-METHOD.
C       THE MATRIX A NEEDS NOT TO BE GIVEN EXPLICITLY.
C       EIGENVALUES OF T APPROXIMATE THE EIGENVALUES OF A.
C       AS EACH STEP OF THE LANCZOS PROCESS NEEDS A MATRIX-VECTOR
C       MULTIPLICATION, THIS PROCESS IS ONLY SUITABLE FOR SPARSE MATRICES.
C ********************
C INPUT-PARAMETERS
C ********************
C   N         -INTEGER. THE ORDER OF THE MATRIX A.
C   IFIRST    -INTEGER. THE FIRST COLUMN OF THE MATRIX T, WHICH HAS TO BE
C                   COMPUTED ON THIS CALL OF LSVLAN.
C                   ON INITIAL CALL IFIRST SHOULD BE 1.
C                   IF LSVLAN IS RESTARTED (SEE INPUT-PARAMETER RESTRT),
C                   IFIRST SHOULD BE EQUAL TO THE LAST COLUMN-NUMBER OF T IN
C                   THE PREVIOUS CALL PLUS 1.
C   M         -INTEGER. THE TOTAL NUMBER OF COLUMNS OF T TO BE COMPUTED
C                   ( THE NUMBER OF COLUMNS IN EARLIER CALLS ARE INCLUDED).
C   RESTRT    -LOGICAL.
C                   RESTRT=.FALSE.  : INITIAL CALL FOR A NEW PROBLEM.
C                   RESTRT=.TRUE.   : INDICATES A RESTART AFTER A PREVIOUS
C                                 CALL OF THE SAME PROBLEM.
C   Q1        -DIMENSION Q1(N). IF RESTRT=.FALSE. : ARBITRARY STARTING VECTOR
C                   FOR THE LANCZOS PROCESS, NOT NECESSARILY OF NORM 1.
C                   IF RESTRT=.TRUE. : Q1(N) SHOULD HAVE THE SAME VALUES AS ON
C                   EXIT OF THE PREVIOUS CALL (ALSO OUTPUT-PARAMETER).
C   AX        -SUBROUTINE AX(Y,AY,N)
C                   DIMENSION Y(N),AY(N)
C                                 THIS USER-SUPPLIED SUBROUTINE DELIVERS FOR A
C                                 GIVEN VECTOR Y THE VECTOR AY, THAT RESULTS FROM
C                                 MULTIPLYING THE MATRIX A WITH THE VECTOR Y.
C                                 Y SHOULD NOT BE DESTROYED WITHIN AX.
C   SAVEQ     -SUBROUTINE SAVEQ(Q,N)
C                   DIMENSION Q(N)
C                                 THIS USER-SUPPLIED SUBROUTINE, WHICH CAN BE USED
C                                 TO STORE THE COLUMNS Q OF THE ORTHOGONAL TRANS-
C                                 FORMATION MATRIX, FOR USE IN COMPUTING THE
C                                 EIGENVECTORS OF A.
C                                 IF EIGENVECTORS ARE NOT DESIRED, THIS SUBROUTINE
C                                 MUST STILL BE SUPPLIED--IT NEED NOT ACTUALLY
C                                 DO ANYTHING.
```

```
C   U           -DIMENSION U(N). SCRATCH-ARRAY.
C   Q0          -DIMENSION Q0(N). IF RESTRT=.FALSE. : Q0 NEED NOT TO BE
C                       INITIALIZED.
C                       IF RESTRT=.TRUE. : Q0 SHOULD HAVE THE SAME VALUES AS ON
C                       EXIT OF THE PREVIOUS CALL OF LSVLAN
C                       ( ALSO OUTPUT-PARAMETER).
C   ALPHA       -DIMENSION ALPHA(M). IF RESTRT=.FALSE. : NO INITIALIZATION
C                       NECESSARY.
C                       IF RESTRT=.TRUE. : ALPHA(1) UP TO ALPHA(MM), WHERE MM
C                       (MM.LT.M) IS THE VALUE OF M IN THE PREVIOUS CALL
C                       OF LSVLAN FOR THE SAME PROBLEM, SHOULD CONTAIN
C                       THE VALUES OF ALPHA AT EXIT OF THIS PREVIOUS CALL
C                       ( ALSO OUTPUT-PARAMETER).
C   BETA        -DIMENSION BETA(M). IF RESTRT=.FALSE. : NO INITIALIZATION
C                       NECESSARY.
C                       IF RESTRT=.TRUE. : BETA(1) UP TO BETA(MM) SHOULD CONTAIN
C                       THE VALUES OF BETA AT EXIT OF THE PREVIOUS CALL.
C                       ( ALSO OUTPUT-PARAMETER).
C *******************
C OUTPUT-PARAMETERS
C *******************
C   Q1          -DIMENSION Q1(N). Q1 CONTAINS THE LAST COLUMN USED IN THE LANCZOS
C                       PROCESS OF THE ORTHOGONAL TRANSFORMATION MATRIX.
C                       ( ALSO INPUT-PARAMETER).
C   Q0          -DIMENSION Q0(N). Q0 CONTAINS THE COLUMN PREVIOUS TO Q1 IN THE
C                       TRANSFORMATION PROCESS. Q0 AND Q1 ARE NECESSARY FOR
C                       RESTART PURPOSES (ALSO INPUT-PARAMETER).
C   ALPHA       -DIMENSION ALPHA(M). THE DIAGONAL OF THE TRIDIAGONAL MATRIX T,
C                       WHICH IS ROUGHLY SIMILAR TO A. ( ALSO INPUT-PARAMETER).
C   BETA        -DIMENSION BETA(M). THE SUPERDIAGONAL ELEMENTS OF THE MATRIX T.
C                       BETA(M) CONTAINS RESTART-INFORMATION.
C                       ( ALSO INPUT-PARAMETER).
C *******************
C INTERNALLY CALLED SUBPROGRAMS
C *******************
C               VVIPP (70923)
C *******************
C REMARKS
C *******************
C   I)          LSVLAN IS SPECIALLY DESIGNED FOR THE DETERMINATION OF THE
C               EXTREME EIGENVALUES OF A SPARSE MATRIX A. IT IS OBSERVED,
C               THAT THE EIGENVALUES OF THE TRIDIAGONAL MATRIX T, REPRESENTED
C               BY ALPHA AND BETA, FOR INCREASING ORDER OF T TEND TO THE FIXED
C               VALUES, WHICH CAN BE CONSIDERED AS EIGENVALUES OF A.
C               FOR PROBLEMS, ARISING WITH THE DETERMINATION OF THE EIGENVALUES
C               WE REFER TO:
C                       VAN KATS J.M., VAN DER VORST H.A.,
C                       "NUMERICAL EXPERIMENTS OF THE PAIGE-STYLE LANCZOS METHOD
C                        FOR THE COMPUTATION OF EXTREME EIGENVALUES OF LARGE
C                        SPARSE MATRICES",
C                       1976, ACCU, TR3.
C   II)         EIGENVALUES OF THE TRIDIAGONAL MATRIX T CAN BE COMPUTED BY
C                       IMTQL1
C                       RATQR
C                       BISECT
C               ALL CONTAINED IN EISPACK ( THIS PACKAGE IS IN UTRECHT AVAILABLE
C               AS LIBRARY ON PERMANENT FILE: LINEARALGEBRA,ID=UACCU).
C   III)        LSVLAN IS SENT IN BY J. LEWIS (COMPUTER SCIENCE DEPARTMENT)
C               STANFORD, U.S.A..
C *******************
C EXAMPLE OF USE
C *******************
C
C         IN THE FOLLOWING PROGRAM 16 LANCZOS-STEPS ARE PERFORMED ON A 21TH
C         ORDER MATRIX (WHICH IS KNOWN AS THE MATRIX W21+ OF WILKINSON).
C         EIGENVALUES OF THE MATRIX W21+ ARE COMPUTED BY EVSCAN, USING
C         THE RESULTS OF LSVLAN.
C
```

```
C                       PROGRAM LNCZOS(OUTPUT)
C                       DIMENSION Q0(21),Q1(21),U(21),ALPHA(16),BETA(16)
C                       LOGICAL RESTRT
C                       EXTERNAL W21PLS,SAVEQ
C                       DIMENSION TK(16,2),TK1(16,2)
C                       LOGICAL LCW,UP,DIV
C                       N=21
C                       IFIRST=1
C                       M=16
C                       RESTRT=.FALSE.
C                       NSEED=110777
C                       CALL RANSET(NSEED)
C                       DO 10 I=1,N
C               10      Q1(I)=RANF(SEED)
C
C                       CALL LSVLAN(N,IFIRST,M,RESTRT,Q1,W21PLS,SAVEQ,U,Q0.
C                      +       ,ALPHA,BETA)
C
C                       MT=16
C
C                       CALL EVSCAN(M,N,MT,ALPHA,BETA,NEV,LOW,UP,DIV,
C                      +        NCIV,TK,TK1,IERR,EPS)
C
C                       IF(IERR.NE.0) STOP "ERROR IN EIGENVALUECCMPUTATION"
C
C                       IF(LOW) PRINT 1000
C                       IF(UP) PRINT 1010
C                       IF(DIV) PRINT 1020,NDIV
C                       PRINT 1030,NEV
C
C               C IF NEV EQUALS 0 NO EIGENVALUE INTERVALS HAVE BEEN DETECTED.
C               C LSVLAN SHOULD BE RESTARTED FOR FURTHER DETECTION, WITH THE
C               C DIMENSIONS OF ALPHA, BETA, TK AND TK1 PROPERLY ADJUSTED.
C
C                       IF (NEV.EQ.0) GOTO 70
C
C                       DO 60 I=1,NEV
C               60      PRINT 1040,I,TK(I,1),TK(I,2)
C
C               70      CONTINUE
C
C               1000    FORMAT(* CONVERGENCE AT LOWER SIDE *)
C               1010    FORMAT(* CONVERGENCE AT UPPER SIDE *,/)
C               1020    FORMAT(* *,I3,* INTERVALS AT LOWER END *)
C               1030    FORMAT(* *,//,* *,I3,* INTERVALS ARE FOUND *,/,
C                      +        *  NR    LOWERBOUND             UPPERBCUND*,/)
C               1040    FORMAT(* *,I3,*  *,2(E20.13,*  *))
C                       END
C
C                       SUBROUTINE SAVEQ(Q,N)
C                       DIMENSION Q(N)
C
C               C NO EIGENVECTORS ARE COMPUTED, SO WE DO NOT STORE
C               C THE ORTHOGONAL TRANSFORMATION-MATRIX.
C
C                       RETURN
C                       END
C
C                       SUBROUTINE W21PLS(X,AX,N)
C                       DIMENSION X(N),AX(N)
C
C               C MATRIX IS TAKEN FROM WILKINSON (THE ALGEBRAIC EIGENVALUEPROBLEM)
C               C AND IS CALLED W21+.
C
C               C      10    1    0
C               C       1    9    1    0
C               C       0    1    8    1    0
C               C             .  .  .  .
C               C                .  .  .  .
C               C              1    8    1    0
C               C              0    1    9    1
C               C                   0    1   10
```

```
C
C             AX(1)=10*X(1)+X(2)
C             DO 10 I=2,20
C             AX(I)=X(I-1)+IABS(11-I)*X(I)+X(I+1)
C     10      CONTINUE
C             AX(21)=X(20)+10*X(21)
C             RETURN
C             END
C
C
C       THE OUTPUT OF THIS PROGRAM IS:
C
C
C             CONVERGENCE AT LOWER SIDE
C             CONVERGENCE AT UPPER SIDE
C
C               2 INTERVALS AT LOWER END
C
C
C               6 INTERVALS ARE FOUND.
C               NR     LOWERBOUND               UPPERBOUND
C
C               1    -.1125441522119E+01     -.1125441522119E+01
C               2     .2538058170973E+00      .2538058170996E+00
C               3     .7003952209422E+01      .7003952209424E+01
C               4     .8038941122376E+01      .8038941122376E+01
C               5     .9210678647337E+01      .9210678647337E+01
C               6     .1074619418290E+02      .1074619418290E+02
C
C ********************
C METHOD
C ********************
C       BASED ON THE PAIGE STYLE LANCZOS PROCESS DESCRIBED IN:
C             LEWIS J., THESIS (TO APPEAR).
C             VAN KATS J.M., VAN DER VORST H.A., "NUMERICAL EXPERIMENTS OF
C                   THE PAIGE-STYLE LANCZOS METHOD FOR THE COMPUTATION OF
C                   EXTREME EIGENVALUES OF LARGE SPARSE MATRICES",
C                   1976, ACCU TR3.
C             VAN KATS J.M., VAN DER VORST H.A., "AUTOMATIC MONITORING OF
C                   LANCZOS SCHEMES FOR SYMMETRIC AND SKEW-SYMMETRIC
C                   GENERALIZED EIGENVALUE PROBLEMS, 1977, ACCU TR7.



      SUBROUTINE LSVLAN(N, FIRST, M, RESTRT, Q1, AX, SAVEQ, U, Q0,
     1 ALPHA, BETA)
C  SUBROUTINE TO CARRY OUT LANCZOS PROCESS FOR A SYMMETRIC REAL MATRIX
C  INPUT:  N => THE SIZE OF THE MATRIX
C          FIRST => THE FIRST COLUMN TO BE COMPUTED ON THIS CALL TO THE
C                   THE SUBROUTINE (ON INITIAL CALL, FIRST WILL BE 1)
C          M => THE MAXIMUM NUMBER OF COLUMNS OF THE PROCESS TO BE
C               COMPUTED -
C          RESTRT => A LOGICAL VARIABLE INDICATING WHETHER WE ARE
C                    RESTARTING FROM A PREVIOUS PARTIAL
C                    TRIDIAGONALIZATION OR STARTING A NEW ONE.
C          Q1 => THE FIRST COLUMN OF Q, A REAL VECTOR, NOT ASSUMED
C                TO BE OF NORM 1.
C          AX => A SUBROUTINE TO CARRY OUT THE MATRIX-VECTOR
C                MULTIPLICATION.  INPUT TO AX IS A REAL VECTOR.
C                OUTPUT SHOULD BE A REAL VECTOR.
C          SAVEQ => A SUBROUTINE WHICH SAVES THE COLUMNS GENERATED BY
C                   THE LANCZOS ALGORITHM, FOR USE IN COMPUTING
C                   EIGENVECTORS.  IF EIGENVECTORS ARE NOT DESIRED,
C                   THIS SUBROUTINE MUST STILL BE SUPPLIED--IT NEED
C                   NOT ACTUALLY DO ANYTHING.
```

```
C  SCRATCH:     U => A REAL      VECTOR OF LENGTH N
C              Q0 => A REAL       VECTOR OF LENGTH N
C                     NEITHER U NOR Q0 NEED BE INITIALIZED
C      THE CONTENTS OF Q1, Q0 AND U WILL BE DESTROYED
C OUTPUT:    ALPHA => THE DIAGONAL OF THE TRIDIAGONAL MATRIX WHICH IS
C                     ROUGHLY SIMILAR TO A.
C            BETA  => THE SUPERDIAGONAL OF THE TRIDIAGONAL MATRIX
C                     ALPHA AND BETA ARE REAL VECTORS.
C  ------------------------------------------------------------------
C  PROCESS USED: THIS SUBROUTINE IMPLEMENTS A REFINED VERSION OF THE
C  PAIGE-STYLE LANCZOS PROCESS.
C  ------------------------------------------------------------------
      INTEGER N, M, FIRST
      LOGICAL RESTRT
      EXTERNAL AX, SAVEQ
      REAL Q1(N), Q0(N)
      REAL U(N)
      REAL ALPHA(M), BETA(M)
C -------------------------------------------------------------------
      REAL LTEMP, LALPHA, LBETA
      REAL EPSLON
      INTEGER I, K
C -------------------------------------------------------------------
C     MACHINE DEPENDENT QUANTITIES:
C         EPSLON IS THE RELATIVE TRUNCATION LEVEL OF SINGLE PRECISION
      DATA EPSLON /1.E-14/
C -----------------------------------------------------------------
      IF (RESTRT) GO TO 60
C ============================================
C I N I T I A L I Z A T I O N
C ============================================
C STEP 1:  NORMALIZE Q1 TO UNIT LENGTH
      LTEMP = VVIPP(Q1,1,Q1,1,N,T)
      IF (LTEMP.EQ.1.0) GO TO 20
      LTEMP = 1.0/SQRT(LTEMP)
      DO 10 I=1,N
         Q1(I) = Q1(I)*LTEMP
   10 CONTINUE
C COMPUTE Q2
   20 CALL SAVEQ(Q1, N)
      CALL AX(Q1, U, N)
      LALPHA = VVIPP(U,1,Q1,1,N,T)
      DO 30 I=1,N
         U(I) = U(I) - LALPHA*Q1(I)
   30 CONTINUE
C FOR USE IN AN ITERATIVE LANCZOS PROCESS, SINCE BETA MAY BE
C VERY SMALL, WE REORTHOGONALIZE Q2 WITH RESPECT TO Q1 IN ORDER
C TO AVOID LOSS OF A LOT OF OUR ORTHOGONALITY RIGHT OFF.
      LTEMP = VVIPP(U,1,Q1,1,N,T)
      DO 40 I=1,N
         U(I) = U(I) - LTEMP*Q1(I)
   40 CONTINUE
C NOW NORMALIZE Q2
      LBETA = VVIPP(U,1,U,1,N,T)
      LBETA = SQRT(LBETA)
      LTEMP = 1.0/LBETA
      DO 50 I=1,N
         Q0(I) = Q1(I)
         Q1(I) = U(I)*LTEMP
   50 CONTINUE
      CALL SAVEQ(Q1, N)
```

```
C COMPUTE INITIAL ALPHA AND BETA.
      ALPHA(1) = LALPHA
      BETA(1) = LBETA
      K = 2
      GO TO 70
C ========================================
C R E S T A R T I N G
C ========================================
C      THE USER MUST HAVE RESTORED Q1 AND Q0, ALPHA, BETA AND
C      KAPPA AS THEY WERE AT THE END OF THE LAST CALL.  LBETA IS
C      STORED IN BETA(FIRST-1)
   60 K = FIRST
      LBETA = BETA(K-1)
      CALL SAVEQ(Q1, M)
C  -------------------------------------------------------------
C  NOW CARRY OUT LANCZOS PROCESS FOR K = 3, 4, ..., M:
C  -------------------------------------------------------------
C FOR K = 2 TO M-1
C          < COMPUTE SUCCEEDING COLUMNS OF ORTHOGONAL MATRIX Q >
   70 IF (K.GT.M) GO TO 100
C COMPUTE PREVIOUS ALPHA AND DIRECTION OF NEXT VECTOR
      LTEMP = VVIPP(Q1,1,Q1,1,N,T)
      CALL AX(Q1, U, N)
      LALPHA = VVIPP(U,1,Q1,1,N,T)
      ALPHA(K) = LALPHA
      DO 80 I=1,N
         U(I) = U(I) - LALPHA*Q1(I) - LBETA*Q0(I)
   80 CONTINUE
C NORMALIZE NEXT VECTOR
      LBETA = VVIPP(U,1,U,1,N,T)
      LBETA = SQRT(LBETA)
      IF (LBETA.LT.EPSLON) LBETA = EPSLON
      LTEMP = 1.0/LBETA
      BETA(K) = LBETA
      DO 90 I=1,N
         Q0(I) = Q1(I)
         Q1(I) = U(I)*LTEMP
   90 CONTINUE
      IF (K.LT.M) CALL SAVEQ(Q1, N)
      K = K + 1
      GO TO 70
  100 RETURN
      END
```

```
C ********************
C HEADING 70518
C ********************
C        SUBROUTINE GENLAN(N,IFIRST,M,RESTRT,Q1,CX,BX,SAVEQ,U,UW,
C                  Q0,ALPHA,BETA,ANTI)
C            LOGICAL RESTRT,ANTI
C            DIMENSION Q1(N),Q0(N),U(N),ALPHA(M),BETA(M),UW(N)
C            EXTERNAL CX,BX,SAVEQ
C ********************
C PURPOSE
C ********************
C        TO GENERATE A TRIDIAGONAL MATRIX T BY THE LANCZOS PROCESS FOR A
C        MATRIX C*B, WHERE C IS WHETHER SYMMETRIC OR SKEW-SYMMETRIC AND
C        B IS SYMMETRIC POSITIVE DEFINITE.
C        THE MATRICES B AND C NEED NOT TO BE GIVEN EXPLICITLY.
C        EIGENVALUES OF T APPROXIMATE THE EIGENVALUES OF C*B.
C        AS EACH STEP OF THE LANCZOS PROCESS NEEDS 2 MATRIX-VECTOR
C        MULTIPLICATIONS, THIS PROCESS IS VERY ATTRACTIVE FOR SPARSE
C        MATRICES.
```

```
C ********************
C INPUT-PARAMETERS
C ********************
C   N          -INTEGER. THE ORDER OF THE MATRICES C AND B.
C   IFIRST     -INTEGER. THE INDEX OF THE FIRST COLUMN OF THE MATRIX T, WHICH
C                      HAS TO BE COMPUTED ON THIS CALL OF GENLAN.
C                      ON INITIAL CALL IFIRST SHOULD BE 1.
C                      IF GENLAN IS RESTARTED (SEE INPUT-PARAMETER RESTRT),
C                      IFIRST SHOULD BE EQUAL TO THE LAST COLUMN-NUMBER OF T IN
C                      THE PREVIOUS CALL PLUS 1.
C   M          -INTEGER. THE TOTAL NUMBER OF COLUMNS OF T TO BE COMPUTED
C                      ( THE NUMBER OF COLUMNS IN EARLIER CALLS ARE INCLUDED).
C   RESTRT     -LOGICAL.
C                      RESTRT=.FALSE.  : INITIAL CALL FOR A NEW PROBLEM.
C                      RESTRT=.TRUE.   : INDICATES A RESTART AFTER A PREVIOUS
C                                CALL OF THE SAME PROBLEM.
C   Q1         -DIMENSION Q1(N). IF RESTRT=.FALSE. : ARBITRARY NONZERO STARTING
C                      VECTOR FOR THE LANCZOS PROCESS, NOT NECESSARILY OF NORM 1.
C                      IF RESTRT=.TRUE. : Q1(N) SHOULD HAVE THE SAME VALUES AS ON
C                      EXIT OF THE PREVIOUS CALL (ALSO OUTPUT-PARAMETER).
C   CX         -SUBROUTINE CX(Y,CY,N)
C                      DIMENSION Y(N),CY(N)
C                                THIS USER-SUPPLIED SUBROUTINE DELIVERS FOR A
C                                GIVEN VECTOR Y THE VECTOR CY, THAT RESULTS FROM
C                                THE MULTIPLICATION OF Y BY THE (SYMMETRIC OR
C                                SKEW-SYMMETRIC) MATRIX C.
C                                Y SHOULD NOT BE DESTROYED WITHIN CX.
C   BX         -SUBROUTINE BX(Y,BY,N)
C                      DIMENSION Y(N),BY(N)
C                                THIS USER-SUPPLIED SUBROUTINE DELIVERS FOR A
C                                GIVEN VECTOR Y THE VECTOR BY, THAT RESULTS FROM
C                                THE MULTIPLICATION OF Y BY THE SYMMETRIC
C                                POSITIVE DEFINITE MATRIX B.
C                                Y SHOULD NOT BE DESTROYED WITHIN BX.
C   SAVEQ      -SUBROUTINE SAVEQ(Q,N)
C                      DIMENSION Q(N)
C                                THIS USER-SUPPLIED SUBROUTINE, WHICH CAN BE USED
C                                TO STORE THE COLUMNS Q OF THE ORTHOGONAL TRANS-
C                                FORMATION MATRIX, FOR USE IN COMPUTING THE
C                                EIGENVECTORS OF C (IN THIS CASE B SHOULD
C                                BE THE IDENTITY MATRIX).
C                                IF EIGENVECTORS ARE NOT DESIRED OR WHEN B IS
C                                NOT THE IDENTITY MATRIX, THIS SUBROUTINE
C                                MUST STILL BE SUPPLIED--IT NEED NOT ACTUALLY
C                                DO ANYTHING.
C   U          -DIMENSION U(N). SCRATCH-ARRAY.
C   UW         -DIMENSION UW(N). SCRATCH-ARRAY.
C   Q0         -DIMENSION Q0(N). IF RESTRT=.FALSE. : Q0 NEED NOT TO BE
C                      INITIALIZED.
C                      IF RESTRT=.TRUE. : Q0 SHOULD HAVE THE SAME VALUES AS ON
C                      EXIT OF THE PREVIOUS CALL OF GENLAN
C                      ( ALSO OUTPUT-PARAMETER).
C   ALPHA      -DIMENSION ALPHA(M). IF RESTRT=.FALSE. : NO INITIALIZATION
C                      NECESSARY.
C                      IF RESTRT=.TRUE. : ALPHA(1) UP TO ALPHA(MM), WHERE MM
C                      (MM.LT.M) IS THE VALUE OF M IN THE PREVIOUS CALL
C                      OF GENLAN FOR THE SAME PROBLEM, SHOULD CONTAIN
C                      THE VALUES OF ALPHA AT EXIT OF THIS PREVIOUS CALL
C                      ( ALSO OUTPUT-PARAMETER).
C   BETA       -DIMENSION BETA(M). IF RESTRT=.FALSE. : NO INITIALIZATION
C                      NECESSARY.
C                      IF RESTRT=.TRUE. : BETA(1) UP TO BETA(MM) SHOULD CONTAIN
C                      THE VALUES OF BETA AT EXIT OF THE PREVIOUS CALL.
C                      ( ALSO OUTPUT-PARAMETER).
C   ANTI       -LOGICAL.IF THE MATRIX C IS SKEW-SYMMETRIC, ANTI SHOULD BE SET
C                      TO .TRUE., IF THE MATRIX C IS SYMMETRIC ANTI SHOULD BE
C                      SET TO .FALSE..
C ********************
C OUTPUT-PARAMETERS
C ********************
C   Q1         -DIMENSION Q1(N). Q1 CONTAINS THE LAST COLUMN USED IN THE LANCZOS
C                      PROCESS OF THE ORTHOGONAL TRANSFORMATION MATRIX.
C                      ( ALSO INPUT-PARAMETER).
```

```
C   Q0        -DIMENSION Q0(N). Q0 CONTAINS THE COLUMN PREVIOUS TO Q1 IN THE
C                     TRANSFORMATION PROCESS. Q0 AND Q1 ARE NECESSARY FOR
C                     RESTART PURPOSES (ALSO INPUT-PARAMETER).
C   ALPHA     -DIMENSION ALPHA(M). THE DIAGONAL OF THE TRIDIAGONAL MATRIX T,
C                     WHICH IS ROUGHLY SIMILAR TO C*B. ( ALSO INPUT-PARAMETER).
C   BETA      -DIMENSION BETA(M). THE SUPERDIAGONAL ELEMENTS OF THE MATRIX T.
C                     BETA(M) CONTAINS RESTART-INFORMATION.
C                     IF C IS SKEW-SYMMETRIC, THEN THE SUBDIAGONAL ELEMENTS
C                     ARE EQUAL BUT OPPOSITE IN SIGN.
C                     ( ALSO INPUT-PARAMETER).
C ********************
C INTERNALLY CALLED SUBPROGRAMS
C ********************
C        VVIPP (70923)
C ********************
C REMARKS
C ********************
C   I)        IF B IS THE IDENTITYMATRIX AND C IS SYMMETRIC, THEN LSVLAN
C             (70517) SHOULD BE PREFERRED BOTH FOR TIME AND STORAGE CONSIDERA-
C             TIONS.
C   II)       GENLAN IS SPECIALLY ATTRACTIVE FOR THE DETERMINATION OF THE
C             EXTREME EIGENVALUES OF A SPARSE MATRIX C*B WITH C AND/OR B SPARSE.
C             THE EIGENVALUES OF THE TRIDIAGONAL MATRIX T, REPRESENTED
C             BY ALPHA AND BETA, FOR INCREASING ORDER OF T TEND TO THE FIXED
C             VALUES, WHICH CAN BE CONSIDERED AS EIGENVALUES OF C*B.
C             FOR PROBLEMS, ARISING WITH THE DETERMINATION OF THE EIGENVALUES
C             WE REFER TO:
C                     VAN KATS J.M., VAN DER VORST H.A.,
C                     "NUMERICAL EXPERIMENTS OF THE PAIGE-STYLE LANCZOS METHOD
C                      FOR THE COMPUTATION OF EXTREME EIGENVALUES OF LARGE
C                      SPARSE MATRICES",
C                     1976, ACCU, TR3.
C   III)      THOSE EIGENVALUES OF THE TRIDIAGONAL MATRIX T, REPRESENTED
C             BY THE ARRAYS ALFHA AND BETA, WHICH CORRESPOND TO EIGENVALUES
C             OF THE ORIGINAL PRODUCT-MATRIX C*B CAN BE COMPUTED BY EVSCAN
C             (70519).
C   IV)       IF THE MATRIX IS SKEW-SYMMETRIC, THEN THE EIGENVALUES DETERMINED
C             BY EVSCAN (70519) SHOULD BE MULTIPLIED BY SQRT(-1).
C   V)        GENLAN IS WRITTEN BY H.A. VAN DER VORST (ACCU) UTRECHT.
C ********************
C EXAMPLE OF USE
C ********************
C
C         THE FOLLOWING PROGRAM COMPUTES, WITH EVSCAN (70519), THE EIGENVALUES
C         OF A SKEW SYMMETRIC MATRIX.
C
C
C                     PROGRAM LNCGEN(OUTPUT)
C                     DIMENSION Q0(20),Q1(20),U(20),UW(20),
C                    *    ALPHA(25),BETA(25)
C                     LOGICAL RESTRT,ANTI,LOW,UP,DIV
C                     EXTERNAL SKEWA,IDENTI,SAVEQ
C                     DIMENSION TK(25,2),TK1(25,2)
C                     N=20
C                     IFIRST=1
C                     M=25
C                     RESTRT=.FALSE.
C                     ANTI=.TRUE.
C                     NSEED=300977
C                     CALL RANSET(NSEED)
C                     DO 10 I=1,N
C               10    Q1(I)=RANF(SEED)
C               C A NONZERO RANDOM STARTVECTOR
C                     CALL GENLAN(N,IFIRST,M,RESTRT,Q1,SKEWA,IDENTI,
C                    *    SAVEQ,U,UW,Q0,ALPHA,BETA,ANTI)
C                     MT=25
C                     CALL EVSCAN(M,N,MT,ALPHA,BETA,NEV,LOW,UP,DIV,
C                    *    NDIV,TK,TK1,IERR,EPS)
```

```
C               IF(IERR.NE.0)STOP "ERROR IN EIGENVALUE COMPUTATION"
C               IF (LOW) PRINT 1000
C               IF (UP) PRINT 1010
C               IF (DIV) PRINT 1020,NDIV
C               PRINT 1030,NEV
C         C   IF NEV EQUALS 0, NO EIGENVALUE INTERVALS HAVE BEEN DETECTED,
C         C   GENLAN SHOULD BE RESTARTED FOR FURTHER INFORMATION, WITH THE
C         C   DIMENSIONS OF ALPHA,BETA,TK AND TK1 PROPERLY ADJUSTED.
C               IF (NEV.EQ.0) GOTO 70
C               DO 60 I=1,NEV
C         60    PRINT 1040,I,TK(I,1),TK(I,2)
C         70    CONTINUE
C         1000  FORMAT(*  CONVERGENCE AT LOWER END *)
C         1010  FORMAT(*  CONVERGENCE AT UPPER END *)
C         1020  FORMAT(* *,I3,* INTERVALS AT LOWER END *)
C         1030  FORMAT(* *,//,* *,I3,* INTERVALS ARE FOUND *,/,
C              *     *   NR    LOWERBOUND           UPPERBOUND *,/)
C         1040  FORMAT(* *,I3,*  *,2(F20.13,*  *))
C               END
C
C
C               SUBROUTINE SAVEQ(Q,N)
C               DIMENSION Q(N)
C         C   NO EIGENVECTORS ARE REQUIRED, SO WE DO NOT STORE THE
C         C   ORTHOGONAL TRANSFORMATION MATRIX.
C               RETURN
C               END
C
C
C               SUBROUTINE SKEWA(X,CX,N)
C               DIMENSION X(N),CX(N)
C         C   THE SUBDIAGONAL ELEMENTS OF THIS SKEW SYMMETRIC MATRIX ARE
C         C   CHOSEN TO BE -1, THE SUPERDIAGONAL ELEMENTS ARE +1 AND ALL
C         C   OTHER ELEMENTS ARE 0.
C               CX(1)=X(2)
C               N1=N-1
C               DO 10 I=2,N1
C         10    CX(I)=X(I+1)-X(I-1)
C               CX(N)=-X(N1)
C               RETURN
C               END
C
C
C               SUBROUTINE IDENTI(X,BX,N)
C               DIMENSION X(N),BX(N)
C         C   THE TRANSFORMATION B IS THE IDENTITY MATRIX IN THIS CASE.
C               DO 10 I=1,N
C         10    BX(I)=X(I)
C               RETURN
C               END
C
C
C     THE OUTPUT OF THIS PROGRAM IS:
C
C
C          CONVERGENCE AT LOWER END
C          CONVERGENCE AT UPPER END
C          20 INTERVALS AT LOWER END
C
C
C          20 INTERVALS ARE FOUND
C           NR    LOWERBOUND             UPPERBOUND
C
C           1      -1.9775616524498      -1.9776616524498
C           2      -1.9111456115717      -1.9111456115716
C           3      -1.8019377358045      -1.8019377358044
C           4      -1.6524775486317      -1.6524775486316
C           5      -1.4661037436593      -1.4661037436593
C           6      -1.2469796037172      -1.2469796037172
C           7       -.9999999999999       -.9999999999999
C           8       -.7306820487327       -.7306820487327
C           9       -.4450418679125       -.4450418679125
C          10       -.1494601871728       -.1494601871728
```

```
C                 11          .1494601871723            .1494601871723
C                 12          .4450418679125            .4450416679125
C                 13          .7306820487326            .7306820487326
C                 14          .9999999999396            .9999999999396
C                 15         1.2469796037170           1.2469796037170
C                 16         1.4661037436592           1.4661037436593
C                 17         1.6524775486313           1.6524775486314
C                 18         1.8019377358040           1.8019377358040
C                 19         1.9111456115715           1.9111456115716
C                 20         1.9776616524493           1.9776616524493
C
C ********************
C METHOD
C ********************
C          THE GENERALIZED LANCZOS SCHEME AS DESCRIBED IN:
C                      VAN KATS J.M., VAN DER VORST H.A.,
C                                  "AUTOMATICAL MONITORING OF GENERALIZED SYMMETRIC
C                                  OR SKEWSYMMETRIC LANCZOS SCHEMES",
C                                  1977, ACCU, UTRECHT, TR7.


      SUBROUTINE GENLAN(N, IFIRST, M, RESTRT, Q1, AX, BX, SAVEQ, U, UW,
     1 Q0, ALPHA, BETA, ANTI)
      LOGICAL ANTI, RESTRT
      EXTERNAL AX, BX, SAVEQ
      REAL Q1(N), Q0(N), U(N), UW(N), ALPHA(M), BETA(M)
      REAL LTEMP, LALPHA, LBETA
      REAL EPSLON
      INTEGER I, K
      DATA EPSLON /1.E-14/
C GENLAN GENERATES ROWS FOR THE PRODUCT MATRIX  BX * AX, WHERE
C BX IS SYMMETRIC AND POSITIVE DEFINITE,
C AX IS WHETHER SYMMETRIC  :  ANTI=.FALSE. ,
C OR ANTISYMMETRIC  : ANTI=.TRUE.
      IF (RESTRT) GO TO 70
      CALL BX(Q1, UW, N)
      LTEMP = VVIPP(Q1,1,UW,1,N,T)
      IF (LTEMP.EQ.1.0) GO TO 20
      LTEMP = 1.0/SQRT(LTEMP)
      DO 10 I=1,N
         Q1(I) = Q1(I)*LTEMP
         UW(I) = UW(I)*LTEMP
   10 CONTINUE
   20 CALL SAVEQ(Q1, N)
      CALL AX(UW, U, N)
      IF (ANTI) GO TO 40
      LALPHA = VVIPP(U,1,UW,1,N,T)
      DO 30 I=1,N
         U(I) = U(I) - LALPHA*Q1(I)
   30 CONTINUE
   40 CONTINUE
      CALL BX(U, UW, N)
      LTEMP = VVIPP(UW,1,Q1,1,N,T)
      DO 50 I=1,N
         U(I) = U(I) - LTEMP*Q1(I)
   50 CONTINUE
      LBETA = VVIPP(UW,1,U,1,N,T)
      LBETA = SQRT(LBETA)
      LTEMP = 1.0/LBETA
      DO 60 I=1,N
         Q0(I) = Q1(I)
         Q1(I) = U(I)*LTEMP
         UW(I) = UW(I)*LTEMP
   60 CONTINUE
```

```
      CALL SAVEQ(Q1, N)
      IF (ANTI) LALPHA = 0.
      ALPHA(1) = LALPHA
      BETA(1) = LBETA
      K = 2
      GO TO 80
   70 K = IFIRST
      LBETA = BETA(K-1)
      CALL SAVEQ(Q1, N)
   80 IF (K.GT.M) GO TO 140
      CALL AX(UW, U, N)
      IF (ANTI) GO TO 100
      LALPHA = VVIPP(UW,1,U,1,N,T)
      ALPHA(K) = LALPHA
      DO 90 I=1,N
         U(I) = U(I) - LALPHA*Q1(I) - LBETA*Q0(I)
   90 CONTINUE
      GO TO 120
  100 ALPHA(K) = 0.
      DO 110 I=1,N
         U(I) = U(I) + LBETA*Q0(I)
  110 CONTINUE
  120 CALL BX(U, UW, N)
      LBETA = VVIPP(U,1,UW,1,N,T)
      LBETA = SQRT(LBETA)
      IF (LBETA.LT.EPSLON) LBETA = EPSLON
      LTEMP = 1.0/LBETA
      BETA(K) = LBETA
      DO 130 I=1,N
         Q0(I) = Q1(I)
         Q1(I) = U(I)*LTEMP
         UW(I) = UW(I)*LTEMP
  130 CONTINUE
      IF (K.LT.M) CALL SAVEQ(Q1, N)
      K = K + 1
      GO TO 80
  140 RETURN
      END
```

```
C ******************
C HEADING 70519
C ******************
C        SUBROUTINE EVSCAN(M,N,MT,ALPHA,BETA,NEV,LOW,UP,DIV,NDIV,TK,
C                          TK1,IERR,EPS)
C             DIMENSION ALPHA(M),BETA(M),TK(MT,2),TK1(MT,2)
C             LOGICAL LOW,UP,DIV
C ******************
C PURPOSE
C ******************
C        TO DESTILLATE A ROW OF DISJUNCT INTERVALS, EACH OF WHICH CONTAINS AT
C        LEAST ONE EIGENVALUE OF A GIVEN TRIDIAGONALMATRIX T, WHICH ARISES
C        IN THE SINGLE-VECTOR LANCZOS PROCESS.
C ******************
C INPUT-PARAMETERS
C ******************
C   M          -INTEGER. THE ORDER OF THE TRIDIAGONAL MATRIX T, AS GENERATED
C                    BY THE LANCZOS-PROCESS.
C                    (M IS EQUAL TO THE NUMBER OF ITERATIONS IN THIS PROCESS).
C   N          -INTEGER. THE ORDER OF THE ORIGINAL MATRIX A, ON WHICH THE
C                    LANCZOS-PROCESS HAS BEEN APPLIED.
C   MT         -INTEGER. THE NUMBER OF ROWS IN THE ACTUAL DECLARATION OF THE
C                    ARRAYS TK AND TK1.
C   ALPHA      -DIMENSION ALPHA(M). THE DIAGONAL OF THE TRIDIAGONAL MATRIX T
C                    AS DELIVERED BY THE LANCZOS-PROCESS. THE ELEMENTS
C                    OF ALPHA ARE NOT ALTERED.
```

```
C   BETA       -DIMENSION BETA(M). BETA(1) THROUGH BETA(M-1) CONTAIN THE
C                       SUPERDIAGONAL ELEMENTS OF THE TRIDIAGONAL MATRIX T.
C                       THE ELEMENTS OF BETA ARE NOT ALTERED.
C   TK1        -DIMENSION TK1(MT,2). A SCRATCH-ARRAY.
C ******************
C OUTPUT-PARAMETERS
C ******************
C   NEV        -INTEGER. THE NUMBER OF DIFFERENT EIGENVALUE INTERVALS, IN
C                       WHICH EIGENVALUES OF THE ORIGINAL MATRIX A ARE CONTAINED.
C   LOW        -LOGICAL. IF LOW IS .TRUE. : CONVERGENCE AT THE LOWER SIDE OF
C                                       THE SPECTRUM.
C                       IF LOW IS .FALSE. : NO CONVERGENCE AT THE LOWER SIDE.
C   UP         -LOGICAL. IF UP IS .TRUE. : CONVERGENCE AT THE UPPER SIDE OF
C                                      THE SPECTRUM.
C                       IF UP IS .FALSE. : NO CONVERGENCE AT THE UPPER SIDE.
C   DIV        -LOGICAL. IF DIV IS .TRUE. : EVSCAN HAS DETERMINED NDIV EIGENVALUE
C                       INTERVALS, WHICH ARE BELIEVED TO REPRESENT THE NDIV
C                       SMALLEST EIGENVALUES OF A (I.E. ALL NDIV SMALLEST
C                       EIGENVALUES OF A HAVE BEEN DISCOVERED).
C                       CAUTION: FROM THE NATURE OF THE LANCZOS-PROCESS IT IS
C                                CLEAR, THAT THIS PARAMETER MAY YIELD WRONG
C                                INFORMATION.
C   NDIV       -INTEGER. SEE DIV.
C   TK         -DIMENSION TK(MT,2). THE PAIRS (TK(1,1),TK(1,2)),
C                       (TK(2,1),TK(2,2)), ... ,(TK(NEV,1),TK(NEV,2))
C                       REPRESENT THE NEV INTERVALS, WHICH CONTAIN EIGENVALUES
C                       OF A.
C   IERR       -INTEGER. ERRORINDICATION.
C                       THE VALUE OF IERR IS SET EQUAL TO AN ERROR COMPLETION CODE.
C                       THE NORMAL COMPLETION CODE IS 0.
C                       IF MORE THAN 30 ITERATIONS ARE REQUIRED TO DETERMINE
C                       AN EIGENVALUE (USING THE SUBROUTINE IMTQL1) OF THE
C                       SYMMETRIC TRIDIAGONAL MATRIX T, IERR IS SET EQUAL TO
C                       THE INDEX OF THE EIGENVALUE FOR WHICH THE FAILURE OCCURS.
C   EPS        -REAL. A MEASURE FOR THE RELATIVE ACCURACY IN THE COMPUTED
C                       EIGENVALUE INTERVALS, AS DELIVERED IN TK.
C ******************
C INTERNALLY CALLED SUBPROGRAMS
C ******************
C       IMTQL1  (A SUBROUTINE FROM THE EISPACK-PACKAGE)
C       MONDIS
C       SCANMP
C       COMINT
C       EPSSPN
C            (ALL THESE ROUTINES ARE INCLUDED IN THIS DECK)
C       MLTPLT  (70520)
C ******************
C REMARKS
C ******************
C   I)         EVSCAN MAY BE CALLED AFTER A CALL OF LSVLAN (70517) OR
C              GENLAN (70518).
C   II)        SOMETIMES EVSCAN DELIVERS AN INTERVAL, WHICH CONTAINS NO
C              EIGENVALUE OF A. HOWEVER IN SUCH A CASE, THERE IS AN EIGENVALUE
C              IN THE NEIGHBOURHOOD OF THAT INTERVAL. IN THESE SITUATIONS IT
C              IS COMMON THAT THE PROCESS YIELDS ALSO THE INTERVAL IN WHICH
C              THE RESPECTIVE EIGENVALUE IS SITUATED. IN ORDER TO IDENTIFY BOTH
C              INTERVALS AS REPRESENTING THE SAME EIGENVALUE, IT IS ADVISED
C              TO USE THE SUBROUTINE MLTPLT (70520) WITH A SLIGHTLY LARGER
C              VALUE OF THE OUTPUTVALUE EPS (10*EPS, SAY):
C                   EPS=10*EPS
C                   CALL MTLFLT(MT,NEV,EPS,TK)
C   III)       FROM THE NATURE OF THE LANCZOS-PROCESS, IT FOLLOWS THAT
C              POSSIBLY SOME EIGENVALUES ARE NOT DISCOVERED AT ALL.
C              IF THE ORIGINAL MATRIX A HAS A MULTIPLET, THAN THIS MULTIPLICITY
C              IS IGNORED.
C   IV)        EVSCAN IS WRITTEN BY J.M. VAN KATS AND H.A. VAN DER VORST
C              (ACCU) UTRECHT.
```

```
C ******************
C EXAMPLE OF USE
C ******************
C
C    IN THE FOLLOWING PROGRAM 16 LANCZOS-STEPS ARE PERFORMED ON A 21TH
C    ORDER MATRIX (WHICH IS KNOWN AS THE MATRIX W21+ OF WILKINSON).
C    EIGENVALUES OF THE MATRIX W21+ ARE COMPUTED BY EVSCAN, USING
C    THE RESULTS OF LSVLAN.
C
C            PROGRAM LNCZOS(OUTPUT)
C            DIMENSION Q0(21),Q1(21),U(21),ALPHA(16),BETA(16)
C            LOGICAL RESTRT
C            EXTERNAL W21PLS,SAVEQ
C            DIMENSION TK(16,2),TK1(16,2)
C            LOGICAL LOW,UP,DIV
C            N=21
C            IFIRST=1
C            M=16
C            RESTRT=.FALSE.
C            NSEED=110777
C            CALL RANSET(NSEED)
C            DO 10 I=1,N
C       10   Q1(I)=RANF(SEED)
C
C            CALL LSVLAN(N,IFIRST,M,RESTRT,Q1,W21PLS,SAVEQ,U,Q0
C           +        ,ALPHA,BETA)
C
C            MT=16
C
C            CALL EVSCAN(M,N,MT,ALPHA,BETA,NEV,LOW,UP,DIV,
C           +        NDIV,TK,TK1,IERR,EPS)
C
C            IF(IERR.NE.0) STOP "ERROR IN EIGENVALUECOMPUTATION"
C
C            IF(LOW) PRINT 1000
C            IF(UP) PRINT 1010
C            IF(DIV) PRINT 1020,NDIV
C            PRINT 1030,NEV
C
C       C IF NEV EQUALS 0 NO EIGENVALUE INTERVALS HAVE BEEN DETECTED,
C       C LSVLAN SHOULD BE RESTARTED FOR FURTHER DETECTION, WITH THE
C       C DIMENSIONS OF ALPHA, BETA, TK AND TK1 PROPERLY ADJUSTED.
C
C            IF (NEV.EQ.0) GOTO 70
C
C            DO 60 I=1,NEV
C       60   PRINT 1040,I,TK(I,1),TK(I,2)
C
C       70   CONTINUE
C
C       1000 FORMAT(* CONVERGENCE AT LOWER SIDE *)
C       1010 FORMAT(* CONVERGENCE AT UPPER SIDE *,/)
C       1020 FORMAT(* *,I3,* INTERVALS AT LOWER END *)
C       1030 FORMAT(* *,//,* *,I3,* INTERVALS ARE FOUND *,/,
C           +    *  NR    LOWERBOUND           UPPERBOUND*,/)
C       1040 FORMAT(* *,I3,*  *,2(E20.13,*  *))
C            END
C
C            SUBROUTINE SAVEQ(Q,N)
C            DIMENSION Q(N)
C
C       C NO EIGENVECTORS ARE COMPUTED, SO WE DO NOT STORE
C       C THE ORTHOGONAL TRANSFORMATION-MATRIX.
C
C            RETURN
C            END
```

```
C
C                     SUBROUTINE W21PLS(X,AX,N)
C                     DIMENSION X(N),AX(N)
C
C              C MATRIX IS TAKEN FROM WILKINSON (THE ALGEBRAIC EIGENVALUEPROBLEM)
C              C AND IS CALLED W21+.
C
C              C      10    1    0
C              C       1    9    1    0
C              C       0    1    8    1    0
C              C              . . . .
C              C                . . . .
C              C               1    8    1    0
C              C               0    1    9    1
C              C                    0    1   10
C
C                     AX(1)=10*X(1)+X(2)
C                     DO 10 I=2,20
C                     AX(I)=X(I-1)+IABS(11-I)*X(I)+X(I+1)
C              10     CONTINUE
C                     AX(21)=X(20)+10*X(21)
C                     RETURN
C                     END
C
C
C          THE OUTPUT OF THIS PROGRAM IS:
C
C
C              CONVERGENCE AT LOWER SIDE
C              CONVERGENCE AT UPPER SIDE
C
C                2 INTERVALS AT LOWER END
C
C
C                6 INTERVALS ARE FOUNC
C                NR     LOWERBOUND             UPPERBOUND
C
C                1    -.1125441522119E+01    -.1125441522119E+01
C                2     .2533058170973E+00     .2533058170986E+00
C                3     .7003952209422E+01     .7003952209424E+01
C                4     .8033941122376E+01     .8033941122376E+01
C                5     .9210678647337E+01     .9210678647337E+01
C                6     .1074619418290E+02     .1074619418290E+02
C
C ******************
C METHOD
C ******************
C          DISTURBANCE OF THE INTERNAL MONOTONY OF THE EIGENVALUES OF SUCCESSIVE
C          TRIDIAGONAL MATRICES, AS DESCRIBED IN:
C                     VAN KATS J.M., VAN DER VORST H.A.,
C                                "AUTOMATICAL MONITORING OF GENERALIZED SYMMETRIC
C                                OR SKEWSYMMETRIC LANCZOS SCHEMES",
C                                1977, ACCU, UTRECHT, TR7.



   SUBROUTINE EVSCAN(M, N, MT, ALPHA, BETA, NEV, LOW, UP, DIV, NDIV,
  1 TK, TK1, IERR, EPS)
   DIMENSION ALPHA(M), BETA(M), TK(MT,2), TK1(MT,2)
   LOGICAL LOW, UP, DIV
   NEV = 0
   LOW = .FALSE.
   UP = .FALSE.
   NDIV = 0
   DO 10 I=2,N
      J = N - I + 1
      TK(J+1,2) = BETA(J)
10 CONTINUE
```

```
      DO 20 I=1,M
         TK(I,1) = ALPHA(I)
         TK1(I,1) = ALPHA(I)
         TK1(I,2) = TK(I,2)
   20 CONTINUE
      M1 = M - 1

C**********************************************************************

      CALL IMTQL1(M1, TK, TK(1,2), IERR)

C**********************************************************************

      IF (IERR.NE.0) RETURN
      TK(M,1) = 0.

C**********************************************************************

      CALL IMTQL1(M, TK1, TK1(1,2), IERR)

C**********************************************************************

      IF (IERR.NE.0) RETURN

C**********************************************************************

      CALL MONDIS(TK, TK1, M1, EPS)

C**********************************************************************

      EPS = FLOAT(N)*AMAX1(EPS,2.**(-47))
C IF ONE WANTS A LARGER VALUE OF EPS, THIS VALUE SHOULD BE INSERTED HERE
      DO 30 I=1,M1
         TK(I,2) = TK(I,1)
         TK1(I,2) = TK1(I,1)
   30 CONTINUE
      TK1(M,2) = TK1(M,1)
      K = M1
      K1 = M

C**********************************************************************

      CALL SCANMP(MT, K, K1, TK, TK1, EPS)

C**********************************************************************

   40 CONTINUE
      EPSOLD = EPS
C**********************************************************************

      CALL COMINT(MT, TK, TK1, K, K1, EPS, LOW, UP, NEV, DIV, NDIV)

C**********************************************************************
      IF (EPS.GT.EPSOLD) GO TO 40
      IF (.NOT.LOW) DIV = .FALSE.
      RETURN
      END
      SUBROUTINE SCANMP(M, K, K1, TK, TK1, EPS)
      DIMENSION TK(M,2), TK1(M,2)
C WITH THE VALUE OF EPS, TWO ROWS (EACH APART) ARE SCANNED FOR MULTIPLET
C THIS MAY DELIVER A NEW VALUE OF EPS AND THE PROCESS IS REPEATED AS
C LONG AS EPS CHANGES.
C THIS RESULTS IN TWO MULTIPLET-FREE ROWS.
C M IS THE NUMBER OF ROWS IN TK AND TK1.
C K IS THE NUMBER OF INTERVALS IN TK.
C K1 IS THE NUMBER OF INTERVALS IN TK1.
   10 CONTINUE
      KOLD = K
      K1OLD = K1
      EPSOLD = EPS
      CALL MLTPLT(M, K, EPSOLD, TK)
      CALL MLTPLT(M, K1, EPS, TK1)
      EPS = AMAX1(EPSOLD,EPS)
      IF ((K.NE.KOLD) .OR. (K1.NE.K1OLD)) GO TO 10
      RETURN
      END
      SUBROUTINE COMINT(M, TK, TK1, K, K1, EPS, LOWER, UPPER, JDEF,
     1 DIV, JDIV)
      DIMENSION TK(M,2), TK1(M,2)
      LOGICAL LOWER, UPPER, DISJCT
      LOGICAL DIV
C COMPARISION OF TWO ROWS OF INTERVALS GIVEN IN TK AND TK1.
C IF AN INTERVAL IN ONE ROW IS CLOSE TO AN INTERVAL IN THE OTHER ROW,
C THAN THE SPAN OF BOTH INTERVALS IS RECORDED IN TK.
C SO THE VALUES OF TK HAVE BEEN OVERWRITTEN.
C LOWER IS .TRUE. : CONVERGENCE AT THE LOWER END OF THE SPECTRUM.
C UP IS .TRUE. : CONVERGENCE AT THE UPPER END OF THE SPECTRUM.
C DIV IS .TRUE. : A HOLE IN THE SPECTRUM IS DETECTED.
C JDEF IS THE NUMBER OF THE COMPUTED EIGENVALUE INTERVALS.
C JDIV GIVES THE NUMBER OF EIGENVALUE INTERVALS AT THE LOWER END.
      DIV = .FALSE.
      IDIS = 0
      LOWER = .FALSE.
      UPPER = .FALSE.
      FOLD = AMIN1(TK1(1,1),TK(1,1))
      EOLD = (1.-SIGN(0.1,FOLD))*FOLD
      FOLD = EOLD
      J = 1
      J1 = 1
      JDEF = 0
   10 CONTINUE
      CALL EPSSFN(TK(J,1), TK(J,2), TK1(J1,1), TK1(J1,2), EPS, E, F,
     1 DISJCT)
      IF (DISJCT) GO TO 50
      IDIS = 0
      IF (J.EQ.1 .OR. J1.EQ.1) LOWER = .TRUE.
      IF (J.EQ.K .OR. J1.EQ.K1) UPPER = .TRUE.
      IF (F.LE.FOLD) GO TO 20
      IF (E.GT.EOLD) JDEF = JDEF + 1
      TK(JDEF,1) = E
      EOLD = E
      TK(JDEF,2) = F
      FOLD = F
   20 IF (TK(J,2).LE.TK1(J1,2)) GO TO 40
   30 CONTINUE
      J = J + 1
      IF (J.LE.K) GO TO 10
      J = J - 1
      J1 = J1 + 1
      IF (J1.GT.K1) GO TO 80
   40 CONTINUE
      J1 = J1 + 1
      IF (J1.LE.K1) GO TO 10
      J1 = J1 - 1
      J = J + 1
      IF (J.GT.K) GO TO 80
   50 CONTINUE
C IF SUCCESSIVELY 6 DISJUNCT INTERVALS ARE MET, A HOLE IN THE SPECTRUM
C IS SUPPOSED.
      IDIS = IDIS + 1
      IF (.NOT.DIV .AND. (IDIS.EQ.6)) GO TO 60
      GO TO 70
   60 JDIV = JDEF
      DIV = .TRUE.
   70 IF (TK1(J1,1).GT.TK(J,1)) GO TO 30
      GO TO 40
   80 CONTINUE
      IF (.NOT.(LOWER .AND. (.NOT.DIV))) RETURN
      DIV = .TRUE.
      JDIV = JDEF
      RETURN
      END
      SUBROUTINE EPSSPN(A, B, C, D, EPS, E, F, DISJCT)
      LOGICAL DISJCT
C IF TWO INTERVALS [A,B] AND [C,D] ARE RELATIVELY CLOSE WITH RESPECT TO
C EPS, [E,F] GIVES THE SPAN OF BOTH, AND DISJCT IS SET TO .FALSE.
C IF THEY ARE NOT CLOSE, DISJCT IS SET TO .TRUE..
      DISJCT = .TRUE.
      EPSS = 0.
      IF (A.LE.C) GO TO 10
      IF (D.GE.A) GO TO 20
      IF (ABS(D-A)/(1.+ABS(D)).LE.EPS) GO TO 20
      RETURN
   10 IF (C.LE.B) GO TO 20
      IF (ABS(C-B)/(1.+ABS(C)).LE.EPS) GO TO 20
      RETURN
   20 CONTINUE
      DISJCT = .FALSE.
      E = AMIN1(A,C)
      F = AMAX1(B,D)
      EPSS = (F-E)/(1.+ABS(E))
      IF (EPSS.GT.EPS) EPS = EPSS
      RETURN
      END
      SUBROUTINE MONDIS(TK, TK1, K, EPS)
      DIMENSION TK(K), TK1(1)
      REAL LOW
C DISTURBANCE OF MONOTONY OF TWO SUCCESSIVE ROWS OF EIGENVALUES GIVEN IN
C AND TK1.
C TK1 HAS ONE EIGENVALUE MORE THAN TK (K+1 AND K RESP.).
      UP = TK(1)
      VAL = TK1(1)
      EPS = 0.0
      IF (VAL.LE.UP) GO TO 10
      EPS = ABS(VAL-UP)/(ABS(VAL)+1.)
   10 CONTINUE
```

AD-A061 729 ACADEMIC COMPUTER CENTER UTRECHT (NETHERLANDS) F/G 12/1
FAST ITERATIVE METHODS FOR LARGE LINEAR SYSTEMS.(U)
AUG 78 H A VORST DA-ERO-75-G-0084

UNCLASSIFIED NL

2 OF 2
AD
A061729

END
DATE
FILMED
2-79
DDC

```
      DO 30 I=2,K
         LOW = UP
         VAL = TK1(I)
         UP = TK(I)
         IF (VAL.GE.LOW) GO TO 20
         EP = ABS(VAL-LOW)/(ABS(VAL)+1.)
         IF (EP.GT.EPS) EPS = EP
   20    IF (VAL.LE.UP) GO TO 30
         EP = ABS(UP-VAL)/(ABS(VAL)+1.)
         IF (EP.GT.EPS) EPS = EP
   30 CONTINUE
      VAL = TK1(K+1)
      IF (VAL.GE.UP) GO TO 40
      EP = ABS(VAL-UP)/(ABS(VAL)+1.)
      IF (EP.GT.EPS) EPS = EP
   40 CONTINUE
C EPS IS THE MAXIMUM OF THE RELATIVE DISTANCE IN CASE OF DISTURBANCE OF
C THE MONOTONY CRITERIUM.
      RETURN
      END
C -----------------------------------------------------------------------
      SUBROUTINE IMTQL1(N, D, E, IERR)
      INTEGER I, J, L, M, N, II, MML, IERR
      REAL D(N), E(N)
      REAL B, C, F, G, P, R, S, MACHEP
C REAL SQRT,ABS,SIGN
C THIS SUBROUTINE IS A TRANSLATION OF THE ALGOL PROCEDURE IMTQL1,
C NUM. MATH. 12, 377-383(1968) BY MARTIN AND WILKINSON,
C AS MODIFIED IN NUM. MATH. 15, 450(1970) BY DUBRULLE.
C HANDBOOK FOR AUTO. COMP., VOL.II-LINEAR ALGEBRA, 241-248(1971).
C THIS SUBROUTINE FINDS THE EIGENVALUES OF A SYMMETRIC
C TRIDIAGONAL MATRIX BY THE IMPLICIT QL METHOD.
C ON INPUT-
C    N IS THE ORDER OF THE MATRIX,
C    D CONTAINS THE DIAGONAL ELEMENTS OF THE INPUT MATRIX,
C    E CONTAINS THE SUBDIAGONAL ELEMENTS OF THE INPUT MATRIX
C      IN ITS LAST N-1 POSITIONS.  E(1) IS ARBITRARY.
C  ON OUTPUT-
C    D CONTAINS THE EIGENVALUES IN ASCENDING ORDER.  IF AN
C      ERROR EXIT IS MADE, THE EIGENVALUES ARE CORRECT AND
C      ORDERED FOR INDICES 1,2,...IERR-1, BUT MAY NOT BE
C      THE SMALLEST EIGENVALUES,
C    E HAS BEEN DESTROYED,
C    IERR IS SET TO
C      ZERO         FOR NORMAL RETURN,
C      J            IF THE J-TH EIGENVALUE HAS NOT BEEN
C                   DETERMINED AFTER 30 ITERATIONS.
C QUESTIONS AND COMMENTS SHOULD BE DIRECTED TO B. S. GARBOW,
C APPLIED MATHEMATICS DIVISION, ARGONNE NATIONAL LABORATORY
C -----------------------------------------------------------------------
C ********** MACHEP IS A MACHINE DEPENDENT PARAMETER SPECIFYING
C            THE RELATIVE PRECISION OF FLOATING POINT ARITHMETIC.
C            **********
       MACHEP = 2.**(-47)
       IERR = 0
       IF (N.EQ.1) GO TO 140
       DO 10 I=2,N
          E(I-1) = E(I)
   10  CONTINUE
```

```
      E(N) = 0.0
      DO 120 L=1,N
         J = 0
C ********** LOOK FOR SMALL SUB-DIAGONAL ELEMENT **********
   20    DO 30 M=L,N
            IF (M.EQ.N) GO TO 40
            IF (ABS(E(M)).LE.MACHEP*(ABS(D(M))+ABS(D(M+1)))) GO TO 40
   30    CONTINUE
   40    P = D(L)
         IF (M.EQ.L) GO TO 80
         IF (J.EQ.30) GO TO 130
         J = J + 1
C ********** FORM SHIFT **********
         G = (D(L+1)-P)/(2.0*E(L))
         R = SQRT(G*G+1.0)
         G = D(M) - P + E(L)/(G+SIGN(R,G))
         S = 1.0
         C = 1.0
         P = 0.0
         MML = M - L
C ********** FOR I=M-1 STEP -1 UNTIL L DO -- **********
         DO 70 II=1,MML
            I = M - II
            F = S*E(I)
            B = C*E(I)
            IF (ABS(F).LT.ABS(G)) GO TO 50
            C = G/F
            R = SQRT(C*C+1.0)
            E(I+1) = F*R
            S = 1.0/R
            C = C*S
            GO TO 60
   50       S = F/G
            R = SQRT(S*S+1.0)
            E(I+1) = G*R
            C = 1.0/R
            S = S*C
   60       G = D(I+1) - P
            R = (D(I)-G)*S + 2.0*C*B
            P = S*R
            D(I+1) = G + P
            G = C*R - B
   70    CONTINUE
         D(L) = D(L) - P
         E(L) = G
         E(M) = 0.0
         GO TO 20
C ********** ORDER EIGENVALUES **********
   80    IF (L.EQ.1) GO TO 100
C ********** FOR I=L STEP -1 UNTIL 2 DO -- **********
         DO 90 II=2,L
            I = L + 2 - II
            IF (P.GE.D(I-1)) GO TO 110
            D(I) = D(I-1)
   90    CONTINUE
  100    I = 1
  110    D(I) = P
  120 CONTINUE
```

```
      GO TO 140
C ********** SET ERROR -- NO CCNVERGENCE TC AN
C                EIGENVALUE AFTER 30 ITERATICNS **********
  130 IERR = L
  140 RETURN
C        ********** LAST CARD OF IMTQL1 **********
C DATE        01/10/76
C  ********END OF DECK**********   IMTQL1
C  ********************************
      END
```

```
C *******************
C HEADING 70520
C *******************
C           SUBROUTINE MLTPLT(M,NINT,EPS,A)
C                 DIMENSION A(M,2)
C *******************
C PURFOSE
C *******************
C     TO REPLACE, IN A ROW OF INTERVALS, THOSE INTERVALS THAT ARE RELATIVELY
C     CLOSE BY THEIR SPAN.
C *******************
C INPUT PARAMETERS
C *******************
C     M          -INTEGER. THE NUMBER OF ROWS IN THE ACTUAL DECLARATION CF THE
C                          ARRAY A.
C     NINT       -INTEGER. THE NUMBER OF INTERVALS GIVEN IN ARRAY A.
C                          (ALSO OUTFUT PARAMETER)
C     EPS        -REAL. IF THE DISTANCE OF TWC SUCCESSIVE INTERVALS IS RELATIVELY
C                          LESS THAN EPS, THEY ARE REPLACED BY THEIR SPAN.
C                          (ALSO OUTFUT PARAMETER)
C     A          -DIMENSION A(M,2). THE PAIRS [A(1,1),A(1,2)], [A(2,1),A(2,2)] .
C                          [A(NINT,1),A(NINT,2)] REPRESENT THE NINT INTERVALS.
C                          THE FOLLCWING CCNDITICNS SHOULD HCLD:
C                          1. A(I,2) .GE. A(I,1)
C                          2. A(I+1,1) .GE. A(I,2)
C                          (ALSO OUTFUT PARAMETER)
C *******************
C OUTFUT PARAMETERS
C *******************
C     NINT       -INTEGER. THE NUMBER OF DISJUNCT INTERVALS IN THE FINAL ROW A.
C                          (ALSO INFUT PARAMETER)
C     EPS        -REAL. THE MAXIMUM OF THE RELATIVE WIDTHS OF THE INTERVALS IN
C                          THE FINAL ROW A, DEFINED BY:
C                          MAX((A(I,2)-A(I,1))/ABS(A(I,1)+1.0))  .
C                          IF THIS VALUE IS SMALLER THAN THE INFUT VALUE OF EPS THEN
C                          EPS IS NOT CHANGED.
C                          (ALSO INFUT PARAMETER)
C     A          -DIMENSION A(M,2). CONTAINS THE RESULTING NINT INTERVALS.
C                          (ALSO INFUT PARAMETER)
C *******************
C INTERNALLY CALLED SUBPROGRAMS
C *******************
C                -
C *******************
C REMARKS
C *******************
C     I)         MLTPLT MAY BE USED AFTER EVSCAN (70519) IN ORDER TO REMOVE
C                POSSIBLY SPURIOUS RESULTS.
C                FOR DETAILS SEE:
C                          VAN KATS J.M., VAN DER VORST H.A.,
C                                       "AUTOMATICAL MONITORING OF GENERALIZED SYMMETRIC
```

```
C                                       OR SKEWSYMMETRIC LANCZOS SCHEMES",
C                                       1977, ACCU, UTRECHT, TR7.
C    II)        MLTPLT HAS BEEN WRITTEN BY J.M. VAN KATS & H.A. VAN DER VORST
C               (ACCU)  -  UTRECHT
C ********************
C EXAMPLE OF USE
C ********************
C
C                     PROGRAM MLTTES(OUTPUT)
C                     DIMENSION A(4,2)
C                     M=4
C                     NINT=4
C                     EPS=1.E-2
C                     A(1,1)=0.3
C                     A(1,2)=0.305
C                     A(2,1)=0.51
C                     A(2,2)=0.52
C                     A(3,1)=0.5205
C                     A(3,2)=0.53
C                     A(4,1)=0.76
C                     A(4,2)=0.77
C                     CALL MLTPLT(M,NINT,EPS,A)
C                     PRINT 1000,NINT,EPS
C               1000  FORMAT(* *,* NUMBER OF FINAL INTERVALS *,I2,/,
C                    *     * EPS *,E9.3,//) .
C                     PRINT 1010
C                     DO 10 I=1,NINT
C               10    PRINT 1020,A(I,1),A(I,2)
C               1010  FORMAT(* *,* LOWERBOUND  UPPERBOUND *)
C               1020  FORMAT(* *,2(E13.6,2X))
C                     END
C
C
C       THE OUTPUT OF THIS PROGRAM IS:
C
C
C                NUMBER OF FINAL INTERVALS  3
C               EPS  .132E-01
C
C
C                LOWERBOUND  UPPERBOUND
C                 .300000E+00    .305000E+00
C                 .510000E+00    .530000E+00
C                 .760000E+00    .770000E+00


      SUBROUTINE MLTPLT(M, NINT, EPS, A)
      DIMENSION A(M,2)
C THE PART A(NINT,2) CONTAINS THE MULTIPLET-INTERVALS.
      EPSM = EPS
      J = 1
C J GIVES THE NUMBER OF THE CURRENT MULTIPLET.
      DO 20 I=2,NINT
         IF (ABS(A(I,1)-A(J,2))/(1.+ABS(A(I,1))).GT.EPS) GO TC 10
         A(J,2) = A(I,2)
C THE UPPERBOUND OF THE J-TH MULTIPLET IS UPDATED.
         EPSMUL = ABS(A(J,2)-A(J,1))/(1.+ABS(A(J,1)))
         IF (EPSMUL.GT.EPSM) EPSM = EPSMUL
         GO TO 20
   10    J = J + 1
         A(J,1) = A(I,1)
         A(J,2) = A(I,2)
   20 CONTINUE
      NINT = J
      EPS = EPSM
      RETURN
      END
```

References

1. J.H. Wilkinson, The Algebraic Eigenvalue Problem,
Oxford university Press, London (1965).

2. W. Kahan and B. Parlett, How far should you go with the
Lanczos process?
in: ed. J.R. Bunch and D.J. Rose, Sparse matrix
computations, Academic Press, New York (1976),
pp. 131-144.

3. J.M. van Kats and H.A. van der Vorst, Numerical Results of the
Paige-style Lanczos method for the computation of
extreme eigenvalues of large sparse matrices,
TR-3, Academic Computer Centre Utrecht (1977).

4. J.H. Wilkinson and C. Reinsch, Linear Algebra,
Springer Verlag, Berlin (1971).

5. O. Widlund, A Lanczos Method for a Class of Non-Hermitian Systems
of Linear Equations,
Courant Institute, New York University (1976).

6. C.C.Paige, Computational Variants of the Lanczos Method for the
Eigenproblem,
J. Inst. Math. Applics. 10 (1972), pp. 373-381.

7. G.H. Golub, Methods for computing eigenvalues of sparse matrix
equations,
Computer Science Dept., Stanford (1974), pp. 125-148.

8. J.B. French, E.C. Halbert, J.B. McGrory and S.S.M. Wong,
Complex Spectroscopy,
Advances in Nuclear Physics, Vol. 3 (1969).

9. J.A. Meijerink and H.A. van der Vorst, An iterative solution
method for linear systems of which the coefficient
matrix is a symmetric M-matrix,
Math. of Comp., Vol. 31 (1977), pp. 148-162.

10. B.T. Smith et al., Matrix Eigensystem Routines - EISPACK
Guide,
Springer Verlag, Berlin (1974).

11. J.G. Lewis, Algorithm for sparse matrix Eigenvalue problems,
Computer Science Dept., Stan-CS-77-595 (1977).

12. A.K. Cline, G.H. Golub and G.W. Platzman, Calculation of
normal modes of oceans using a Lanczos method.
in: ed. J.R. Buch and D.J. Rose, Sparse matrix
computations, Academic Press, New York (1976),
pp. 409-426.

13. G.W. Platzman, Normal Modes of the Atlantic and Indian Oceans,
J. Phys. Oceanography 5 (1975), pp. 201-221.

14. R.S. Varga, Matrix iterative analysis,
Prentice Hall, Englewood Cliffs, New Jersey (1962).

k	number of eigen-values	$\tilde{\varepsilon}= \frac{\varepsilon}{40*2^{-47}}$	$\frac{\tilde{\varepsilon}}{k}$	k	number of eigen-values	$\tilde{\varepsilon}= \frac{\varepsilon}{40*2^{-47}}$	$\frac{\tilde{\varepsilon}}{k}$
				21	40	152.	7.2
				22	40	132.	6.0
3	0	1	0.3	23	40	276.	12.0
4	24 [1])	5.1	1.3	24	40	188.	7.8
5	32 [1])	21.	4.2	25	40	156.	6.2
6	35 [1])	20.	3.3	26	40	159.	6,1
7	40	13.	1.8	27	40	287.	10.6
8	40	44.	5.5	28	40	275.	9.8
9	40	33.	3.7	29	41 [2])	224.	7.7
10	40	62.	6.2	30	41 [2])	114.	3.8
11	40	56.	5.1	31	40	262.	8.5
12	40	130.	11.0	32	40	282.	8.8
13	40	93.	7.2	33	40	281.	8.5
14	40	72.	5.1	34	40	192.	5.6
15	40	117.	7.8	35	40	359.	10.2
16	40	116.	7.3	36	40	167.	4.6
17	40	61.	3.6	37	42 [2])	388.	10.5
18	40	183.	10.2	38	40	346.	9.1
19	40	205.	10.8	39	40	228.	5.8
20	40	288.	14.4	40	40	215.	5.4

Table I

Results of the monitoring process for the bar problem.

[1]) for k=4,5,6, all eigenvalues were detected at the upper end of the spectrum.

[2]) if the recommended extra scan is performed, 40 eigenvalues are detected.

-1.125441522119	6.000217522256
0.2538058170974	6.000234031583
0.9475343675298	7.003951798616
1.789321352695	7.003952209528
2.130209219363	8.038941115813
2.961058884186	8.038941122828
3.043099292579	9.210678647304
3.996048201383	9.210678647360
4.004354023440	10.74619418290339
4.999782477742	10.74619418290332
5.000244425001	

Table II: Eigenvalue of W^{+}_{21}

± 10.74619418290
± 9.210678647331
± 8.038941119304
± 7.003952002664
± 6.000225680184
± 5.000008158672
± 4.000000205070
± 3.000000003808
± 2.000000000054
± 1.000000000000
0

Table III: Eigenvalues of W^{-}_{21}

m	number of eigenvalues	eps	$\frac{\text{eps}}{21*2^{-47}}$
13	0	.15E-12	1
16	6	.13E-11	8.9
32	20	.83E-12	5.5
150	19	.19E-10	129.
300	19	.16E-10	109.

Table IVa: Results of EVSCAN after m iterations of the Lanczos-scheme applied to W_{21}^{+}.

m	number of eigenvalues	eps	$\frac{\text{eps}}{21*2^{-47}}$
13	0	.15E-12	1
16	0	.15E-12	1
32	21	.15E-12	20.9
150	21	.31E-11	93.
300	21	.29E-10	196.

Table IVb: Results of EVSCAN after m iterations of the Lanczos-scheme applied to W_{21}^{-}.

No	lowerbound	upperbound	upperbound-lowerbound
1	-.1125441522119E+01	-.1125441522119E+01	.135E-12
2	.2538058170973E+00	.2538058170981E+00	.815E-12
3	.7003952209096E+01	.7003952209100E+01	.381E-11
4	.8038941119703E+01	.8038941119703E+01	.114E-12
5	.9210678647329E+01	.9210678647329E+01	.568E-13
6	.1074619418290E+02	.1074619418290E+02	0.

Table Va: m=16

No	lowerbound	upperbound	upperbound-lowerbound
1	-.1125441522119E+01	-.1125441522118E+01	.104E-11
2	.2538058170977E+00	.2538058170980E+00	.293E-12
3	.9475343675301E+00	.9475343675303E+00	.263E-12
4	.1789321352695E+01	.1789321352695E+01	.284E-13
5	.2130209219363E+01	.2130209219363E+01	.711E-13
6	.2961058884185E+01	.2961058884185E+01	.156E-12
7	.3043099292578E+01	.3043099292578E+01	.568E-13
8	.3996048201383E+01	.3996048201383E+01	.114E-12
9	.4004354023441E+01	.4004354023441E+01	0.
10	.4999782477742E+01	.4999782477742E+01	0.
11	.5000244425001E+01	.5000244425001E+01	.853E-13
12	.6000217522256E+01	.6000217522256E+01	.171E-12
13	.6000234031583E+01	.6000234031584E+01	.853E-13
14	.7003951798615E+01	.7003951798615E+01	.256E-12
15	.7003952209528E+01	.7003952209528E+01	.114E-12
16	.8038941115813E+01	.8038941115813E+01	.568E-13
17	.8038941122828E+01	.8038941122828E+01	.114E-12
18	.9210678647304E+01	.9210678647304E+01	0.
19	.9210678647360E+01	.9210678647360E+01	0.
20	.1074619418290E+02	.1074619418290E+02	.125E-11

Table Vb: m=32

No	lowerbound	upperbound	upperbound- lowerbound
1	-.112544l522118E+01	-.1125441522111E+01	.714E-11
2	.2538058170894E+00	.2538058171016E+00	.122E-10
3	.9475343675312E+00	.9475343675369E+00	.575E-11
4	.1789321352696E+01	.1789321352700E+01	.442E-11
5	.2130209219362E+01	.2130209219364E+01	.205E-11
6	.2961058884184E+01	.2961058884186E+01	.189E-11
7	.3043099292578E+01	.3043099292586E+01	.794E-11
8	.3996048201379E+01	.3996048201384E+01	.493E-11
9	.4004354023435E+01	.4004354023442E+01	.614E-11
10	.4999782477739E+01	.4999782477743E+01	.350E-11
11	.5000244424997E+01	.5000244425001E+01	.415E-11
12	.6000217522252E+01	.6000217522256E+01	.449E-11
13	.6000234031577E+01	.6000234031583E+01	.631E-11
14	.7003951798611E+01	.7003951798763E+01	.152E-09
15	.7003952209506E+01	.7003952209528E+01	.218E-10
16	.8038941115805E+01	.8038941115813E+01	.767E-11
17	.8038941122821E+01	.8038941122828E+01	.682E-11
18	.9210678647298E+01	.9210678647360E+01	.619E-10
19	.1074619418289E+02	.1074619418289E+02	.483E-11

Table Vc: m=150

No	lowerbound	upperbound	upperbound- lowerbound
1	-.1125441522119E+01	-.1125441522106E+01	.128E-10
2	.2538058170901E+00	.2538058171041E+00	.140E-10
3	.9475343675327E+00	.9475343675446E+00	.119E-10
4	.1789321352690E+01	.1789321352702E+01	.126E-10
5	.2130209219362E+01	.2130209219372E+01	.958E-11
6	.2961058884136E+01	.2961058884187E+01	.510E-10
7	.3043099292577E+01	.3043099292591E+01	.139E-10
8	.3996048201372E+01	.3996048201384E+01	.112E-10
9	.4004354023434E+01	.4004354023441E+01	.685E-11
10	.4999782477737E+01	.4999782477745E+01	.867E-11
11	.5000244424993E+01	.5000244425001E+01	.801E-11
12	.6000217522246E+01	.6000217522256E+01	.101E-10
13	.6000234031570E+01	.6000234031583E+01	.132E-10
14	.7003951798607E+01	.7003951798614E+01	.747E-11
15	.7003952209507E+01	.7003952209527E+01	.203E-10
16	.8038941115798E+01	.8038941115813E+01	.149E-10
17	.8038941122814E+01	.8038941122828E+01	.136E-10
18	.9210678647290E+01	.9210678647359E+01	.697E-10
19	.1074619418288E+02	.1074619418288E+02	.750E-11

Table Vd: m=300

Eigenvalue intervals determined by EVSCAN after m Lanczos iterations applied to W_{21}^{+}.

No	lowerbound (a)	upperbound (b)	b-a
1	-.1074619418290E+02	-.1074619418290E+02	.248E-11
2	-.9210678647332E+01	-.9210678647332E+01	.227E-12
3	-.8038941119305E+01	-.8038941119305E+01	.114E-12
4	-.7003952002662E+01	-.7003952002662E+01	.256E-12
5	-.6000225680184E+01	-.6000225680183E+01	.142E-12
6	-.5000008158671E+01	-.5000008158671E+01	.227E-12
7	-.4000000205070E+01	-.4000000205070E+01	.227E-12
8	-.3000000003807E+01	-.3000000003806E+01	.384E-12
9	-.2000000000054E+01	-.2000000000054E+01	.185E-12
10	-.1000000000000E+01	-.9999999999998E+00	.298E-12
11	.7531604183839E-12	.1089622219482E-11	.336E-12
12	.1000000000001E+01	.1000000000001E+01	.391E-12
13	.2000000000053E+01	.2000000000053E+01	.284E-12
14	.3000000003807E+01	.3000000003807E+01	.568E-13
15	.4000000205068E+01	.4000000205068E+01	.568E-13
16	.5000008158671E+01	.5000008158671E+01	.199E-12
17	.6000225680181E+01	.6000225680182E+01	.171E-12
18	.7003952002661E+01	.7003952002661E+01	.142E-12
19	.8038941119301E+01	.8038941119301E+01	0.
20	.9210678647327E+01	.9210678647327E+01	.568E-13
21	.1074619418290E+02	.1074619418290E+02	.210E-11

Table VIa: m=32

No	lowerbound (a)	upperbound (b)	b-a
1	-.1074619418290E+02	-.1074619418289E+02	.144E-10
2	-.9210678647332E+01	-.9210678647319E+01	.131E-10
3	-.8038941119305E+01	-.8038941119291E+01	.138E-10
4	-.7003952002662E+01	-.7003952002653E+01	.901E-11
5	-.6000225680184E+01	-.6000225680176E+01	.782E-11
6	-.5000008158671E+01	-.5000008158662E+01	.892E-11
7	-.4000000205070E+01	-.4000000205060E+01	.955E-11
8	-.3000000003808E+01	-.3000000003799E+01	.904E-11
9	-.2000000000055E+01	-.2000000000046E+01	.918E-11
10	-.1000000000002E+01	-.9999999999924E+01	.952E-11
11	-.1337897763203E-11	.4887256470544E-11	.623E-11
12	.9999999999981E+00	.1000000000002E+01	.352E-11
13	.2000000000050E+01	.2000000000054E+01	.355E-11
14	.3000000003803E+01	.3000000003806E+01	.315E-11
15	.4000000205063E+01	.4000000205067E+01	.406E-11
16	.5000008158665E+01	.5000008158667E+01	.222E-11
17	.6000225680176E+01	.6000225680182E+01	.585E-11
18	.7003952002654E+01	.7003952002660E+01	.560E-11
19	.8038941119292E+01	.8038941119314E+01	.221E-10
20	.9210678647315E+01	.9210678647319E+01	.381E-11
21	.1074619418288E+02	.1074619418289E+02	.995E-11

Table VIb: m=150

No	lowerbound (a)	upperbound (b)	b-a
1	-.10746194182 90E+02	-.1074619418287E+02	.335E-10
2	-.9210678647331E+01	-.9210678647302E+01	.290E-10
3	-.8038941119305E+01	-.8038941119280E+01	.254E-10
4	-.7003952002662E+01	-.7003952002643E+01	.196E-10
5	-.6000225680184E+01	-.6000225680158E+01	.261E-10
6	-.5000008158671E+01	-.5000008158651E+01	.206E-10
7	-.4000000205070E+01	-.4000000204931E+01	.140E-09
8	-.3000000003809E+01	-.3000000003788E+01	.207E-10
9	-.2000000000057E+01	-.2000000000036E+01	.208E-10
10	-.1000000000004E+01	-.9999999999878E+00	.160E-10
11	-.3882333782009E-11	.1472361028863E-10	.186E-10
12	.9999999999943E+00	.1000000000003E+01	.855E-11
13	.2000000000046E+01	.2000000000053E+01	.726E-11
14	.3000000003798E+01	.3000000003806E+01	.800E-11
15	.4000000204992E+01	.4000000205063E+01	.710E-10
16	.5000008158657E+01	.5000008158664E+01	.648E-11
17	.6000225680167E+01	.6000225680187E+01	.202E-10
18	.7003952002645E+01	.7003952002653E+01	.821E-11
19	.8038941119280E+01	.8038941119290E+01	.105E-10
20	.9210678647299E+01	.9210678647308E+01	.875E-11
21	.1074619418286E+01	.1074619418288E+02	.148E-10

Table VIc: m=300

Eigenvalue intervals determined by EVSCAN after m Lanczos iterations applied to $\bar{W}_{21}$.

lowerbound (a)	upperbound (b)	b-a
.1001000000000E+01	.1001000000000E+01	.782E-13
.1019000000000E+01	.1019000000000E+01	.924E-13
.2000000000000E+01	.2000000000000E+01	.483E-12
.3039999999996E+01	.3039999999998E+01	.134E-11

Table VIIa: m=40 eps=3.3*40*2^{-47}
see section 4.3

lowerbound (a)	upperbound (b)	b-a
.1001000000000E+01	.1001000000000E+01	.142E-13
.1002000000000E+01	.1002000000000E+01	.711E-14
.1003000000000E+01	.1003000000000E+01	.426E-13
.1004000000000E+01	.1004000000000E+01	.213E-13
.1005000000000E+01	.1005000000000E+01	.497E-13
.1006000000000E+01	.1006000000000E+01	.426E-13
.1007000000000E+01	.1007000000000E+01	.853E-13
.1008000000000E+01	.1008000000000E+01	.426E-13
.1009000000000E+01	.1009000000000E+01	.711E-14
.1010000000000E+01	.1010000000000E+01	.426E-13
.1011000000000E+01	.1011000000000E+01	.284E-13
.1012000000000E+01	.1012000000000E+01	.284E-13
.1013000000000E+01	.1013000000000E+01	.426E-13
.1014000000000E+01	.1014000000000E+01	.213E-13
.1015000000000E+01	.1015000000000E+01	.355E-13
.1016000000000E+01	.1016000000000E+01	.497E-13
.1017000000000E+01	.1017000000000E+01	.213E-13
.1018000000000E+01	.1018000000000E+01	.355E-13
.1019000000000E+01	.1019000000000E+01	.142E-13
.1999999999999E+01	.2000000000000E+01	.696E-12
.3020999999999E+01	.3021000000000E+01	.142E-13
.3022000000000E+01	.3022000000000E+01	.284E-13
.3022999999999E+01	.3023000000000E+01	.284E-13
.3024000000000E+01	.3024000000000E+01	.284E-13
.3024999999999E+01	.3024999999999E+01	.284E-13
.3026000000000E+01	.3026000000000E+01	.284E-13
.3027000000000E+01	.3027000000000E+01	.426E-13
.3028000000000E+01	.3028000000000E+01	0.
.3028999999999E+01	.3028999999999E+01	.142E-13
.3029999999999E+01	.3030000000000E+01	.568E-13
.3030999999999E+01	.3030999999999E+01	.568E-13
.3031999999999E+01	.3031999999999E+01	.426E-13
.3032999999999E+01	.3032999999999E+01	.284E-13
.3033999999999E+01	.3033999999999E+01	.426E-13
.3034999999999E+01	.3034999999999E+01	.142E-13
.3035999999999E+01	.3035999999999E+01	.711E-13
.3036999999999E+01	.3036999999999E+01	.284E-13
.3037999999999E+01	.3037999999999E+01	.142E-13
.3038999999999E+01	.3038999999999E+01	.142E-13
.3039999999999E+01	.3040000000000E+01	.142E-13

Table VIIb: m=45 eps=4.7*40*2^{-47}
see section 4.3

lowerbound	upperbound
-.180316926294 3E+03	-.180316926294 3E+03
-.175759203021 8E+03	-.175759203021 8E+03

Table VIIIa: $\varepsilon=519*2^{-47}$, 40 Lanczos steps.

CP-time required for LSVLAN: 152 seconds
CP-time required for EVSCAN: 0.4 seconds

lowerbound	upperbound
-.180316926294 3E+03	-.180316926294 3E+03
-.175759203021 8E+03	-.175759203021 8E+03
-.174396362300 6E+03	-.174396362300 6E+03
-.173896229622 5E+03	-.173896229622 5E+03
-.173578723826 2E+03	-.173578723826 2E+03
-.173244426157 0E+03	-.173244426157 0E+03
-.173123025077 3E+03	-.173123025077 3E+03
-.172351627059 4E+03	-.172351627059 4E+03
-.157277657764 9E+03	-.157277657764 6E+03
-.156976768752 7E+03	-.156976768752 7E+03

Table VIIIb: $\varepsilon=519*2^{-47}*2.2$, 60 Lanczos steps.

CP-time required for LSVLAN: 226 seconds
CP-time required for EVSCAN: 0.9 seconds

lowerbound	upperbound
-.180316926294 3E+03	-.180316922505 0E+03
-.175759203021 8E+03	-.175759202984 7E+03
-.174396362300 6E+03	-.174396362300 6E+03
-.173896229622 5E+03	-.173896229622 5E+03
-.173578723826 2E+03	-.173578723826 2E+03
-.173244426157 0E+03	-.173244426157 0E+03
-.173123025077 3E+03	-.173123025077 3E+03
-.172351627059 4E+03	-.172351627059 4E+03
-.172013619120 5E+03	-.172013619097 6E+03
-.171436174935 2E+03	-.171436148593 5E+03
-.171164800512 1E+03	-.171164612505 7E+03
-.157798543273 1E+03	-.157798446546 1E+03
-.157701360717 2E+03	-.157700989314 5E+03
-.157570422953 4E+03	-.157570389972 5E+03
-.157433694699 9E+03	-.157433694000 3E+03
-.157277657764 9E+03	-.157277657764 6E+03
-.156976768752 7E+03	-.156976768752 7E+03

Table VIIIc: $=519*10^{-8}$, 60 Lanczos steps

General information:

- order of the matrix n=519.
- the matrix was available on a diskfile, no attempts have been made to optimize the matrix-vector multiplications required for LSVLAN.

m	number of eigenvalues	$\frac{\text{eps}}{992 * 2^{-47}}$	m	number of eigenvalues	$\frac{\text{eps}}{992 * 2^{-47}}$
30	6	6.7	220	56	16.
40	8	2.7	230	59	16.
50	10	2.9	240	62	19.
60	14	2.0	250	64	13.
70	20	2.8	260	65	15.
80	23	7.4	270	67	17.
90	26	6.2	280	68	15.
100	28	3.4	290	68	18.
110	30	6.4	300	73	20.
120	32	5.0	310	77	28.
130	36	11.	320	77	11.
140	37	8.4	330	77 [1]	116. [1]
150	38	6.9	340	78	19.
160	41	9.4	350	83	19.
170	46	12.5	360	85	10.
180	46	7.6	370	87	14.
190	49	4.9	380	87	20.
200	56	16.	390	88	17.
210	56	13.	400	89 [1]	96. [1]

Table IX

Results of EVSCAN after m Lanczos-iterations applied to the 992-th order matrix described in section 4.5

[1] Results after second scan with 5*eps

lowerbound	upperbound
-.2383672694904E+00	-.2383672694901E+00
.2383672694900E+00	.2383672694904E+00

Table Xa: m=15

lowerbound	upperbound
-.2383672694904E+00	-.2383672694904E+00
-.1476900716874E+00	-.1476900716874E+00
-.1373857686379E+00	-.1373857686379E+00
.1373857686379E+00	.1373857686379E+00
.1476900716874E+00	.1476900716874E+00
.2383672694903E+00	.2383672694903E+00

Table Xb: m=30

lowerbound	upperbound
-.2383672694903E+00	-.2383672694903E+00
-.1476900716874E+00	-.1476900716874E+00
-.1373857686379E+00	-.1373857686378E+00
-.1055710572521E+00	-.1055710572521E+00
-.1017762399231E+00	-.1017762399231E+00
-.8098832373012E-01	-.8098832373012E-01
-.8014875412532E-01	-.8014875412531E-01
-.7608597598440E-01	-.7608597598440E-01
.7608597598434E-01	.7608597598435E-01
.8014875412526E-01	.8014875412526E-01
.8098832373009E-01	.8098832373009E-01
.1017762399230E+00	.1017762399230E+00
.1055710572520E+00	.1055710572520E+00
.1373857686378E+00	.1373857686378E+00
.1476900716873E+00	.1476900716873E+00
.2383672694902E+00	.2383672694904E+00

Table Xc: m=60

Eigenvalue intervals delivered by EVSCAN after m Lanczos-iterations applied to the product matrix described in 4.6.

Note: The values in the table above should be multiplied by i(=SQRT(-1)) so that they represent eigenvalues of A.

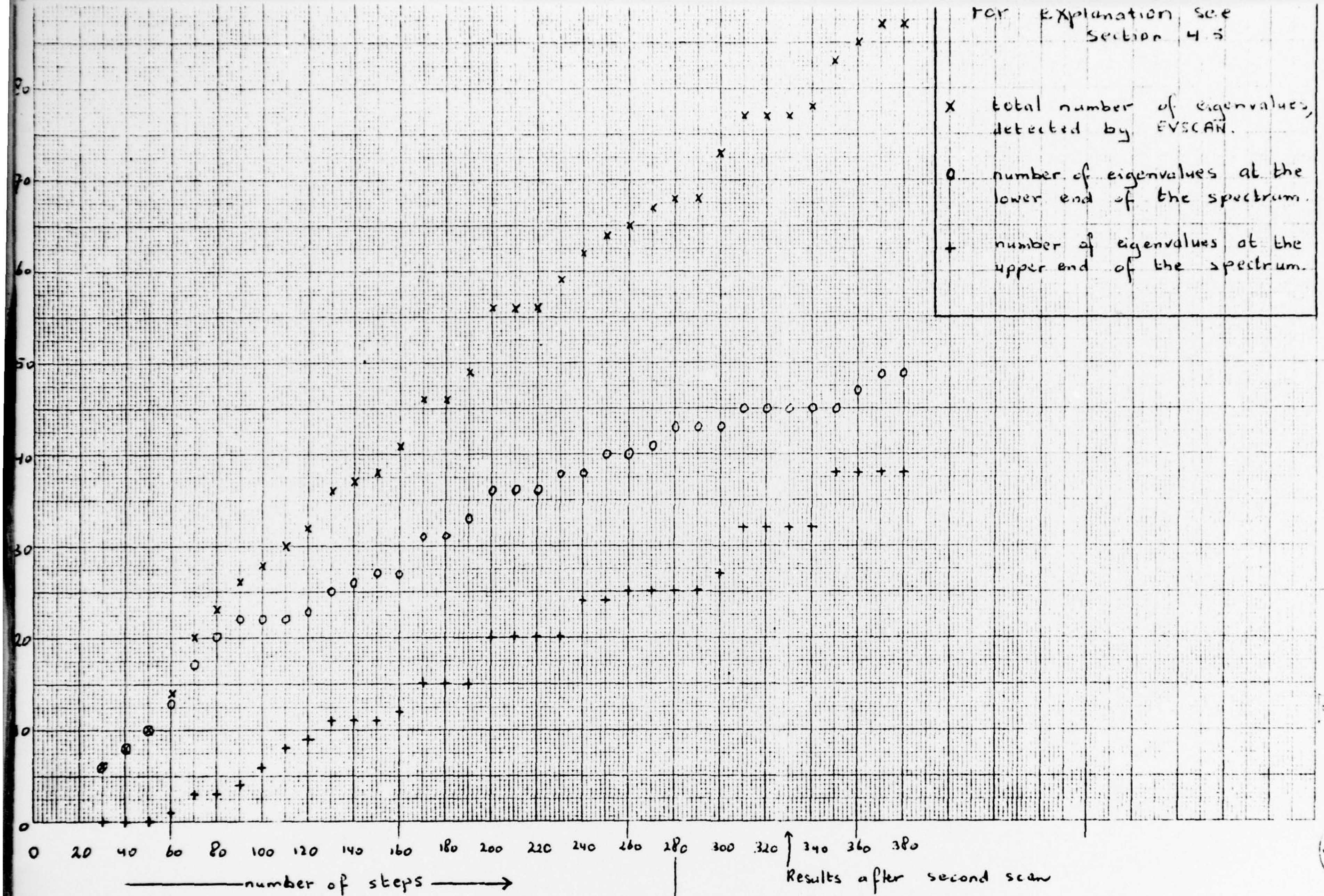

For Explanation see Section 4.5
x total number of eigenvalues, detected by EVSCAN.
o number of eigenvalues at the lower end of the spectrum.
+ number of eigenvalues at the upper end of the spectrum.
0
10
20
30
40
50
60
70
80
0
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
320
340
360
380
number of steps
Results after second scan

Figure I